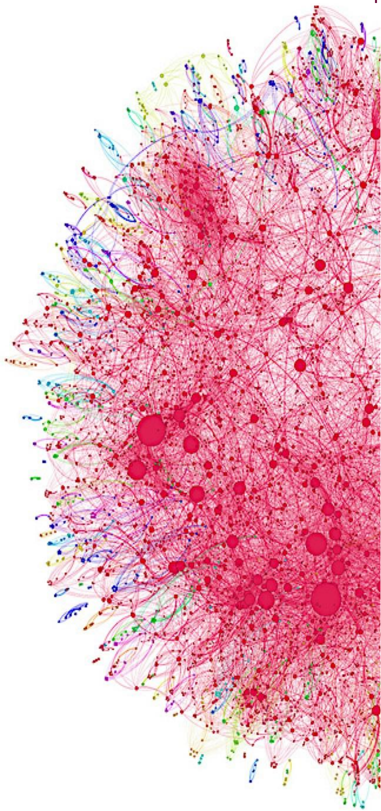
sGradd: Towards RELIABLE S.t..r...e....ami..ng Graph Analytics

Aida Sheshbolouki

Supervisor: Prof. M. Tamer Ozsu

David R. Cheriton School of Computer Science, University of Waterloo

UNIVERSITY OF WATERLOO | DSg Data Systems Group

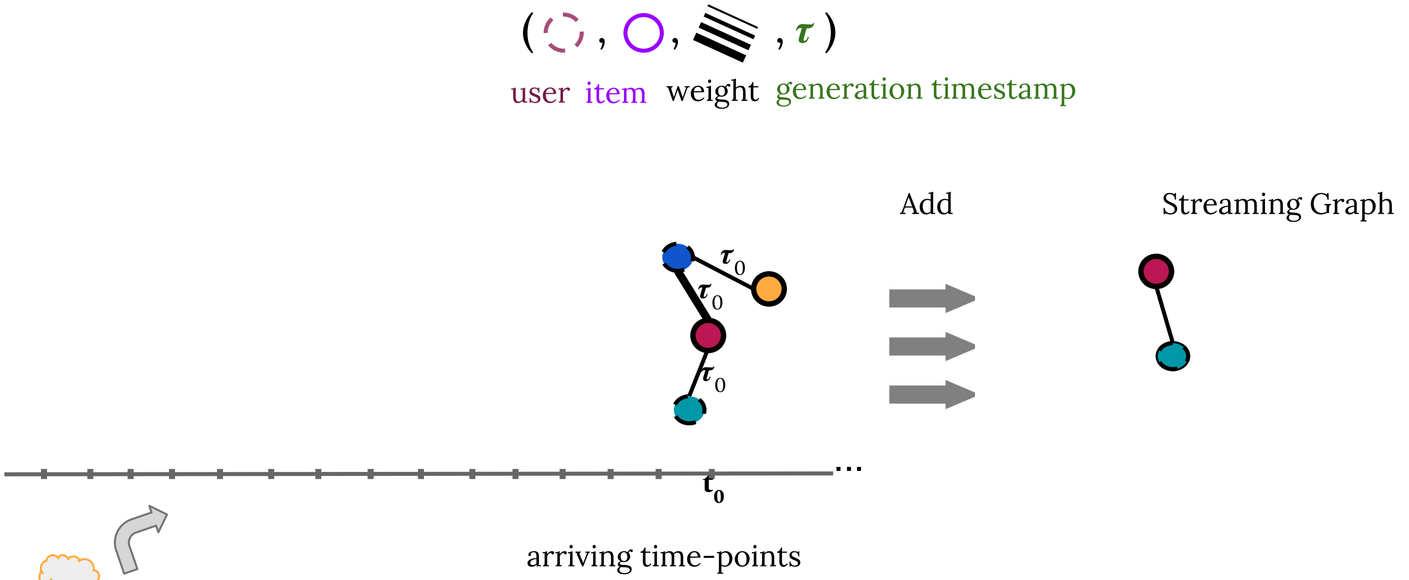


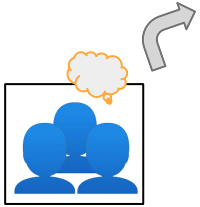
Graphs are everywhere!

Finance Transportation Social Media Use Communications

Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records

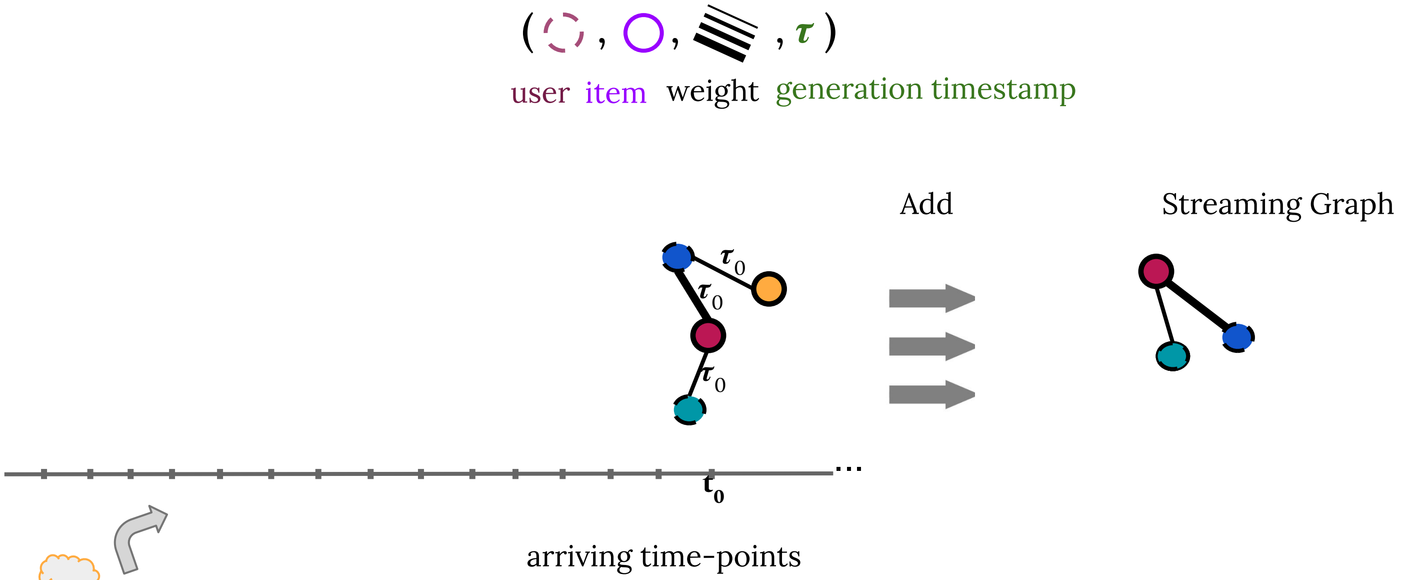


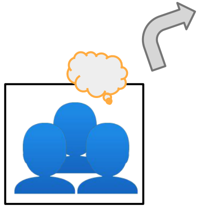


Finance Transportation Social Media Use Communications

Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records

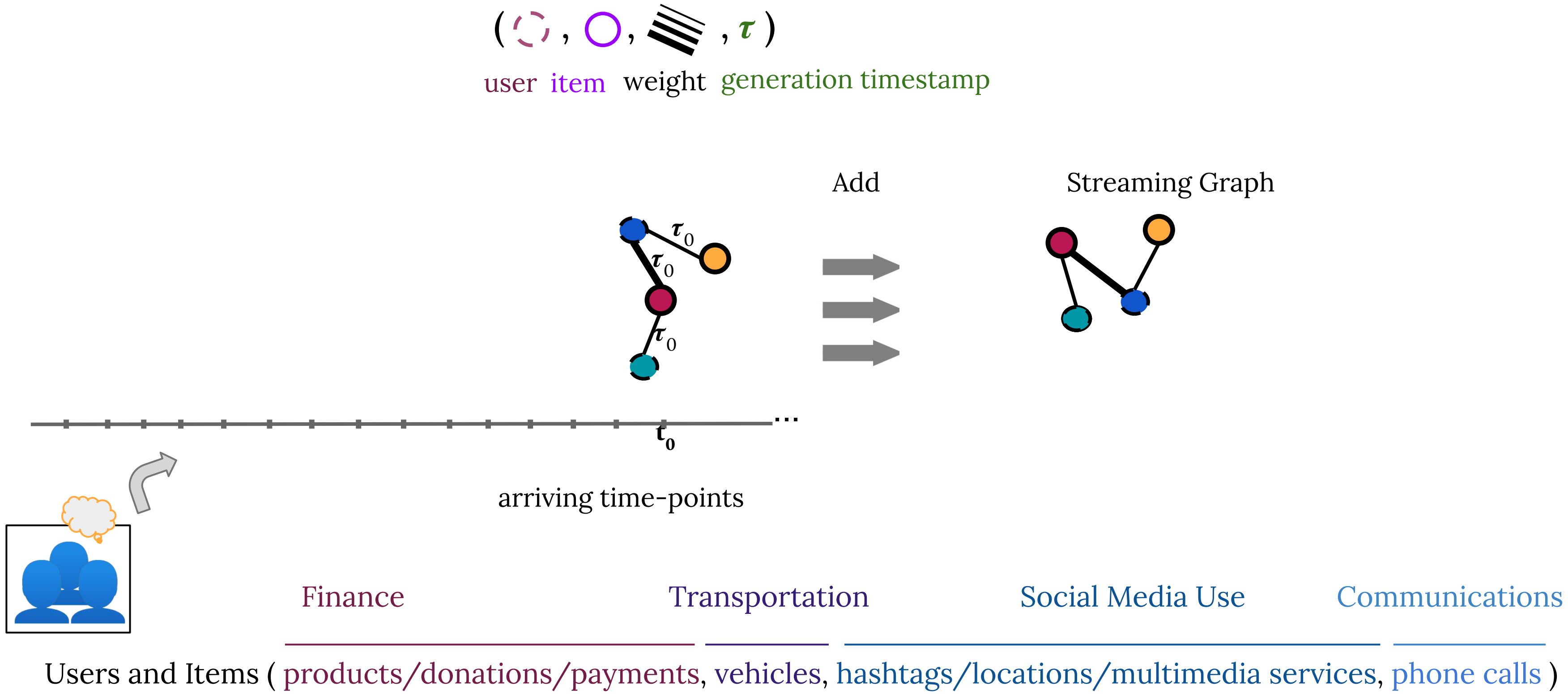




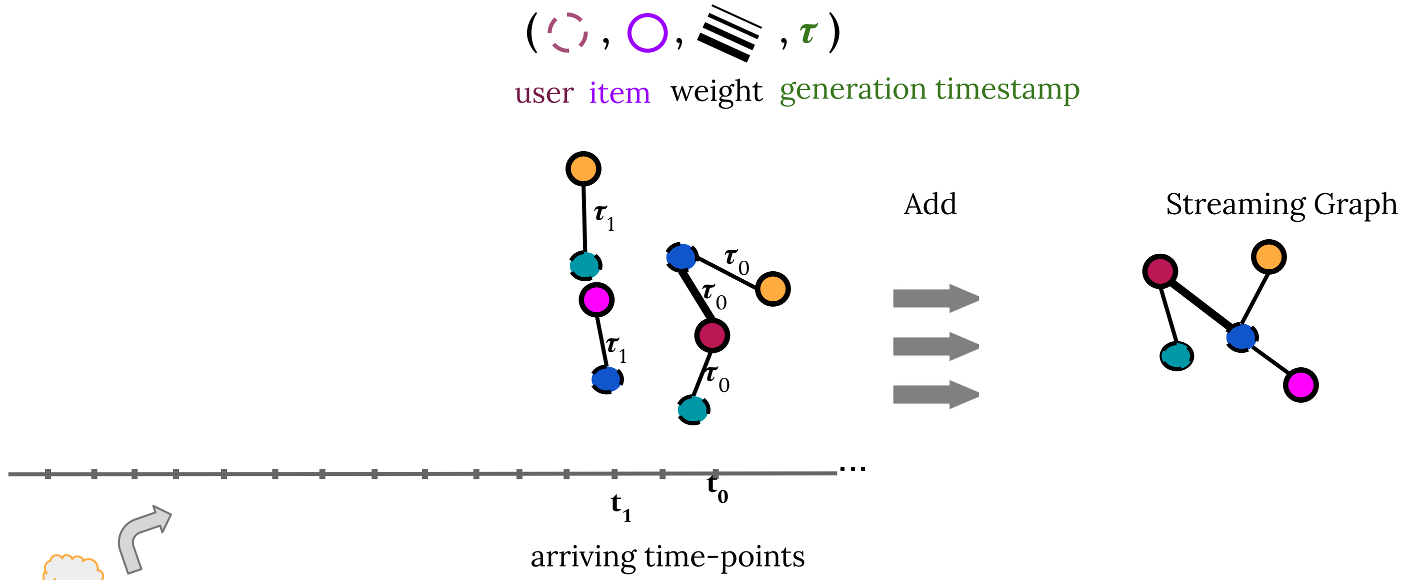
Finance Transportation Social Media Use Communications

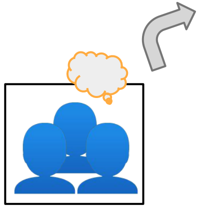
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records



An unbounded stream of linked data records

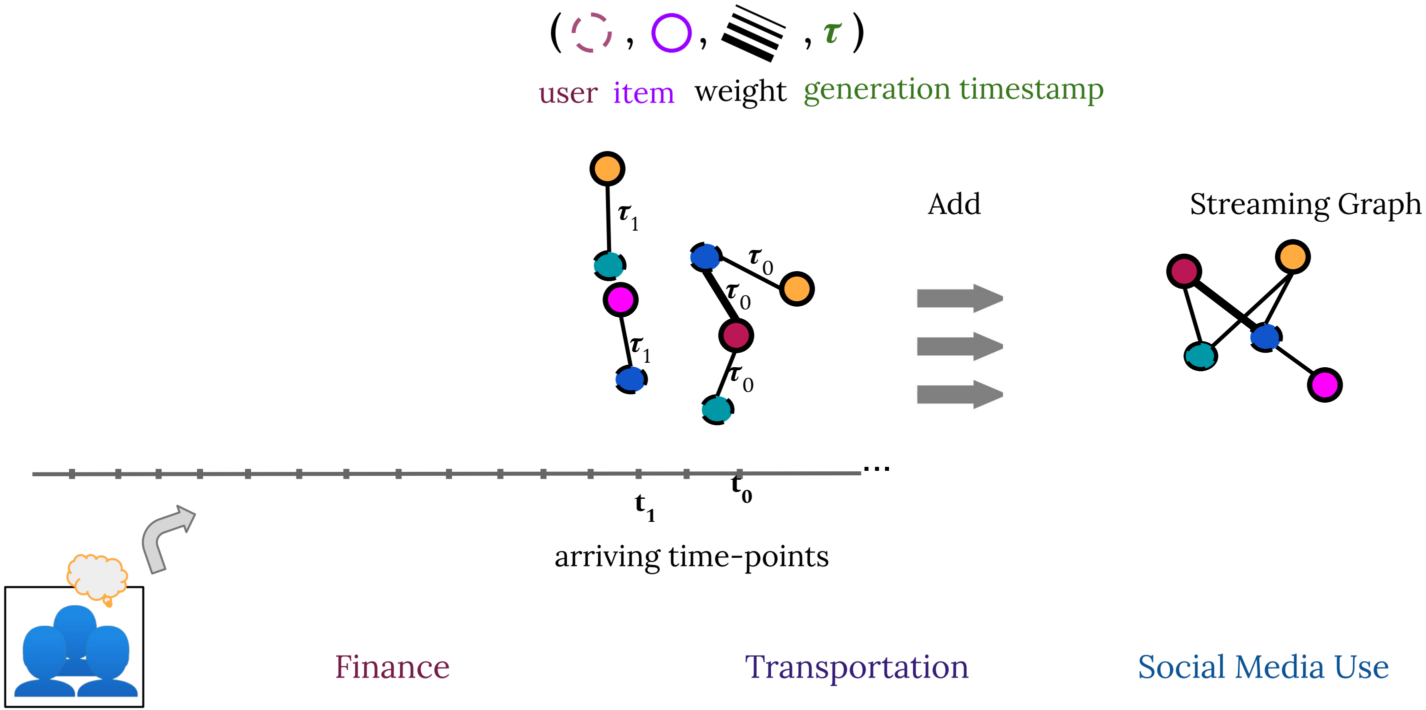




Finance Transportation Social Media Use Communications

Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

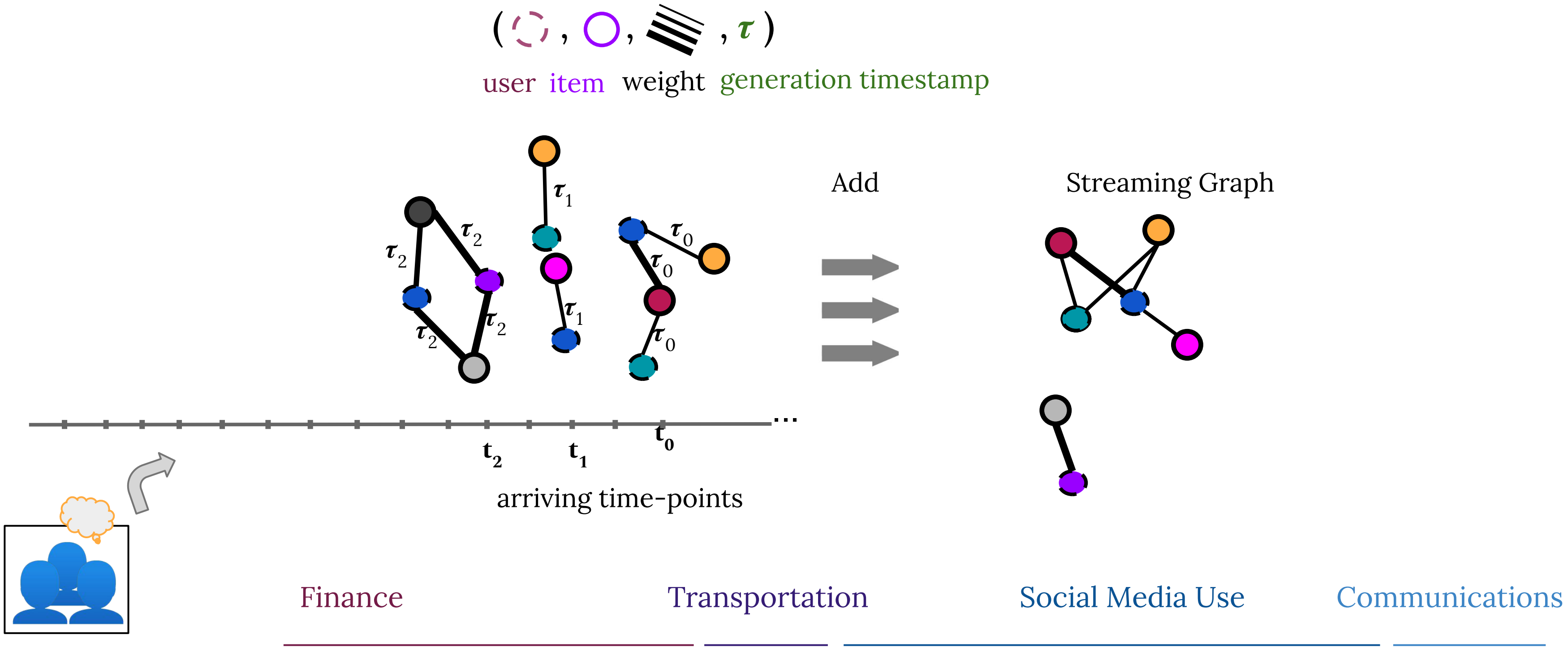
An unbounded stream of linked data records



Finance Transportation Social Media Use Communications

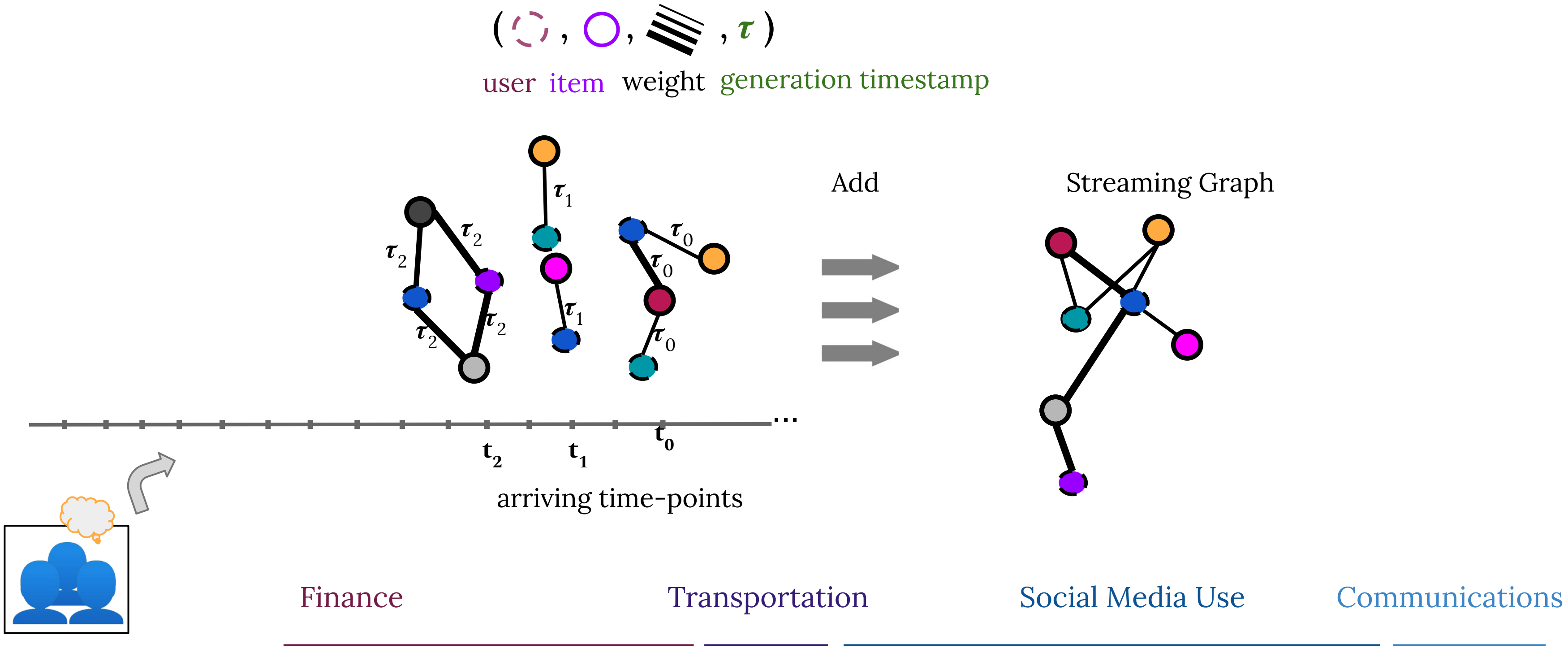
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records



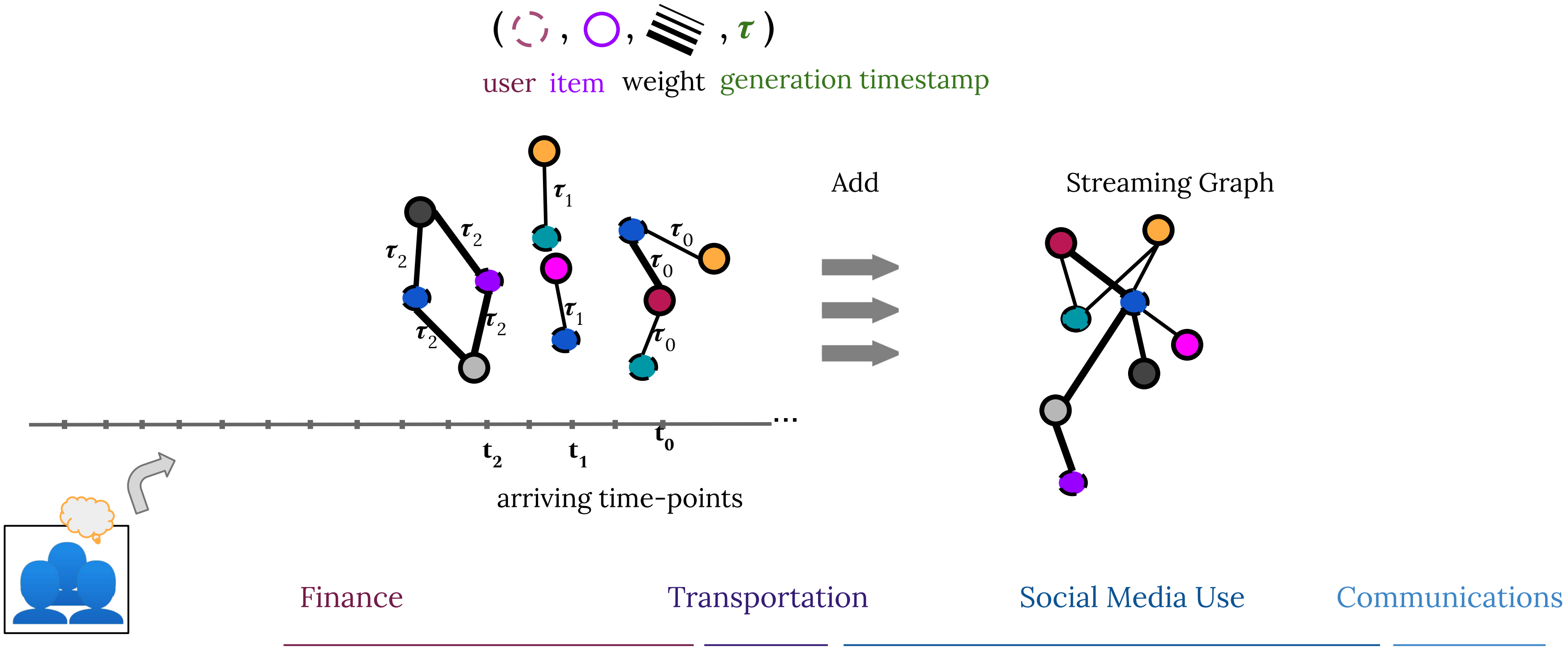
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records



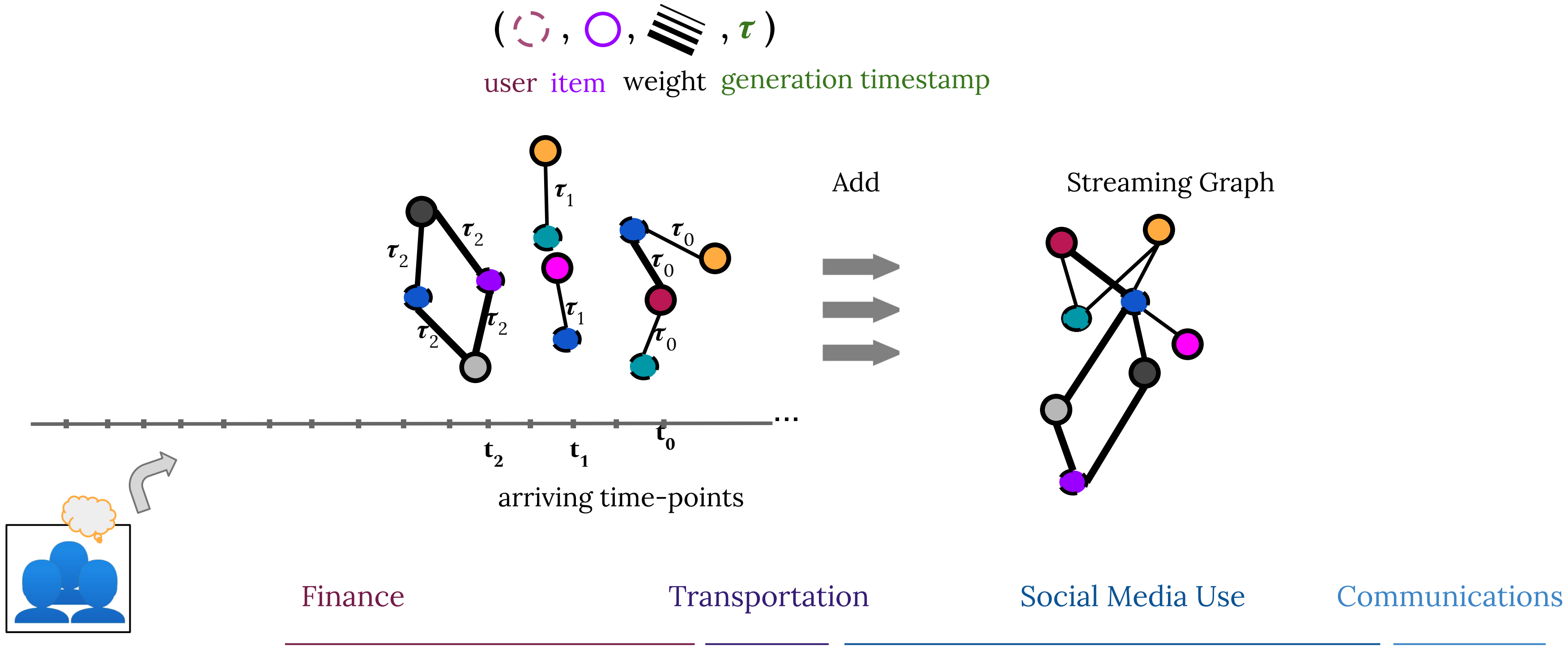
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records

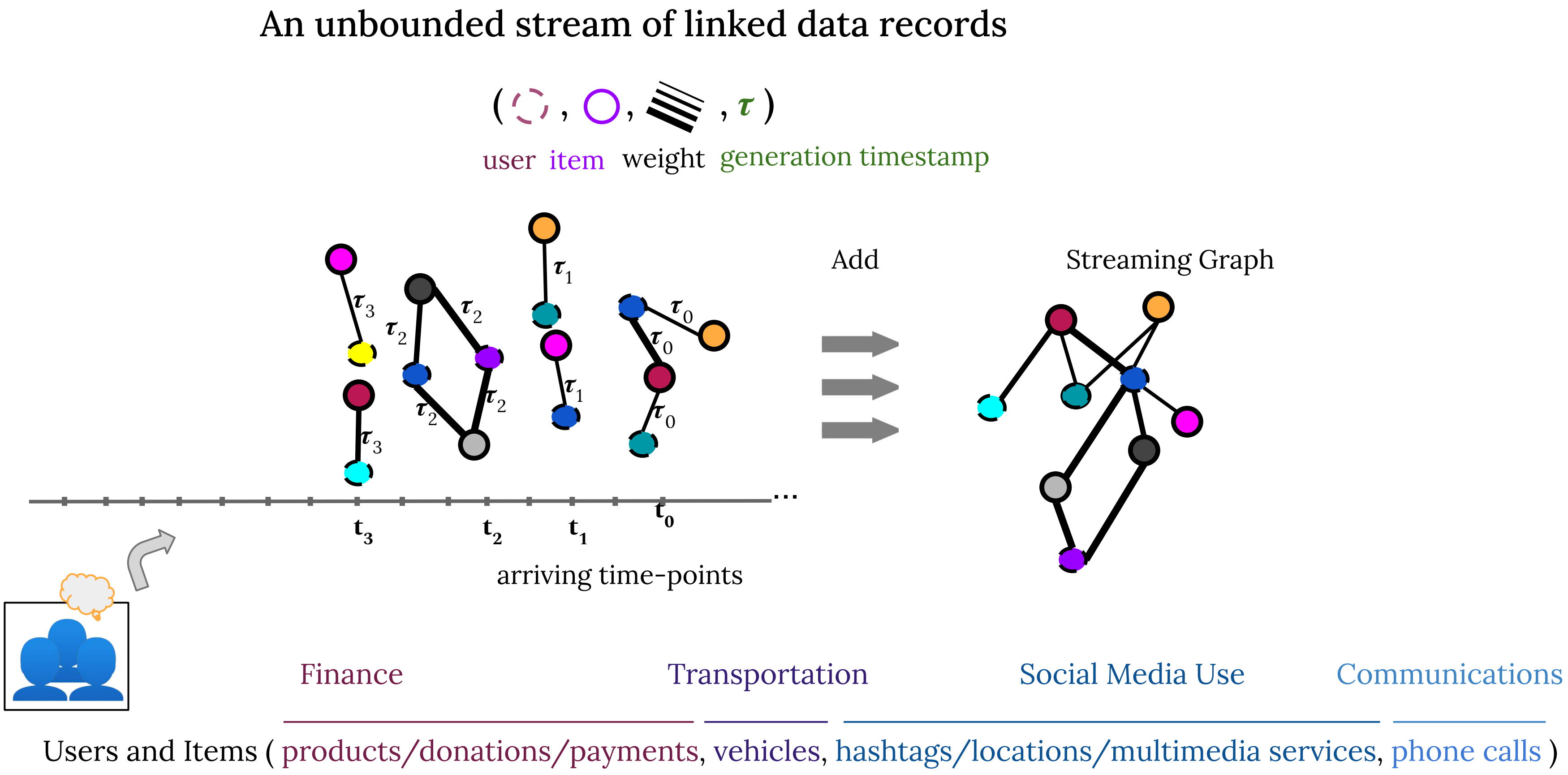


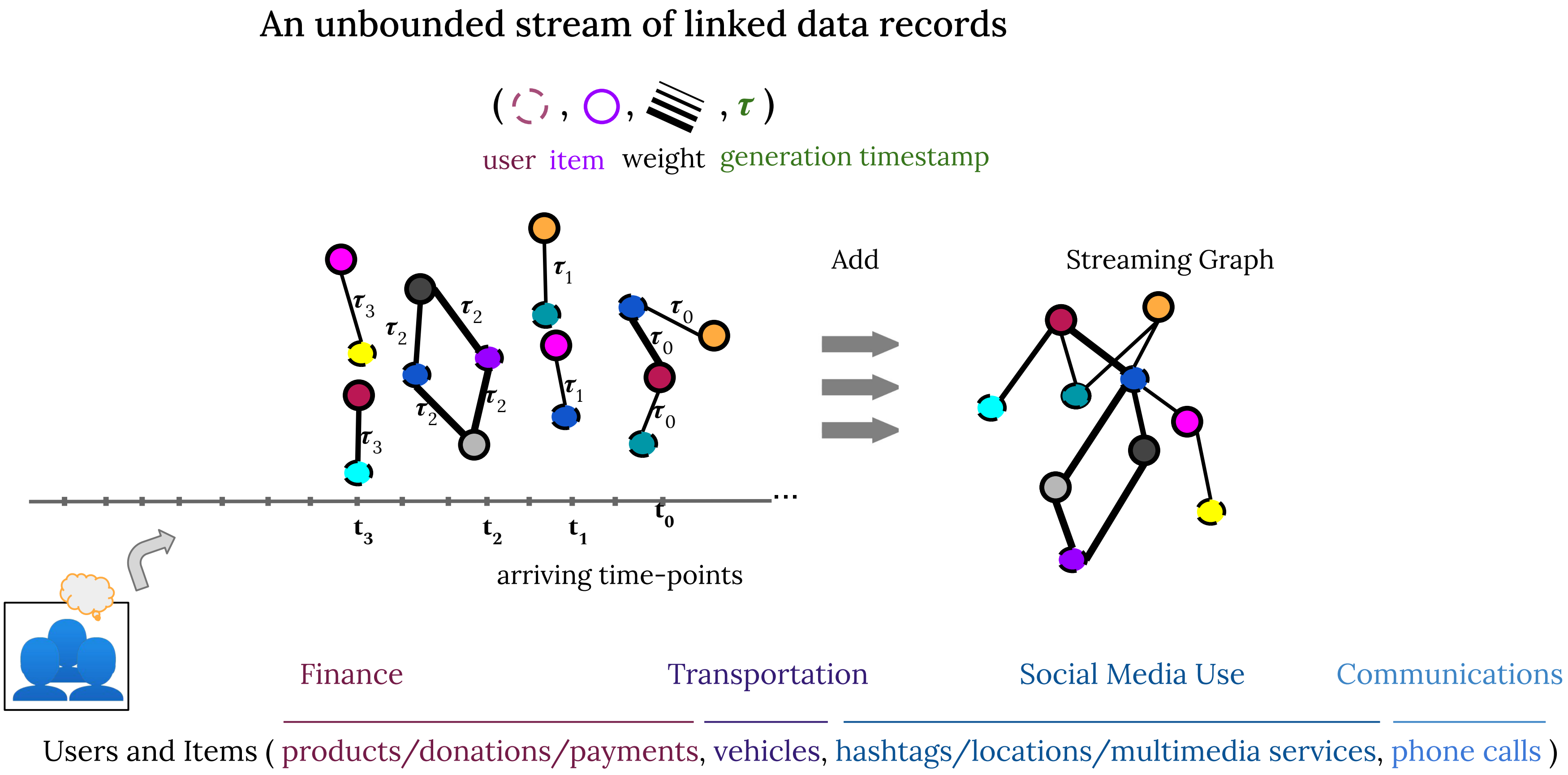
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records

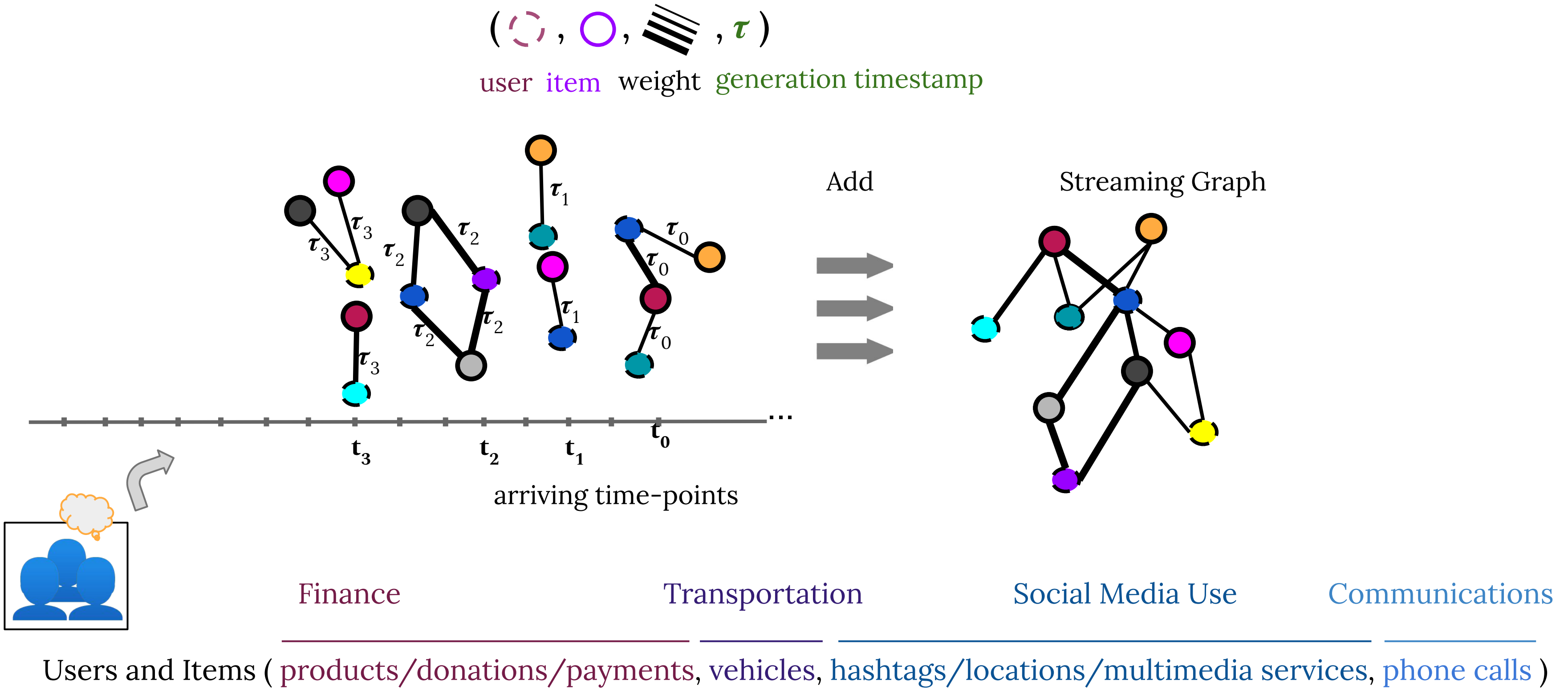


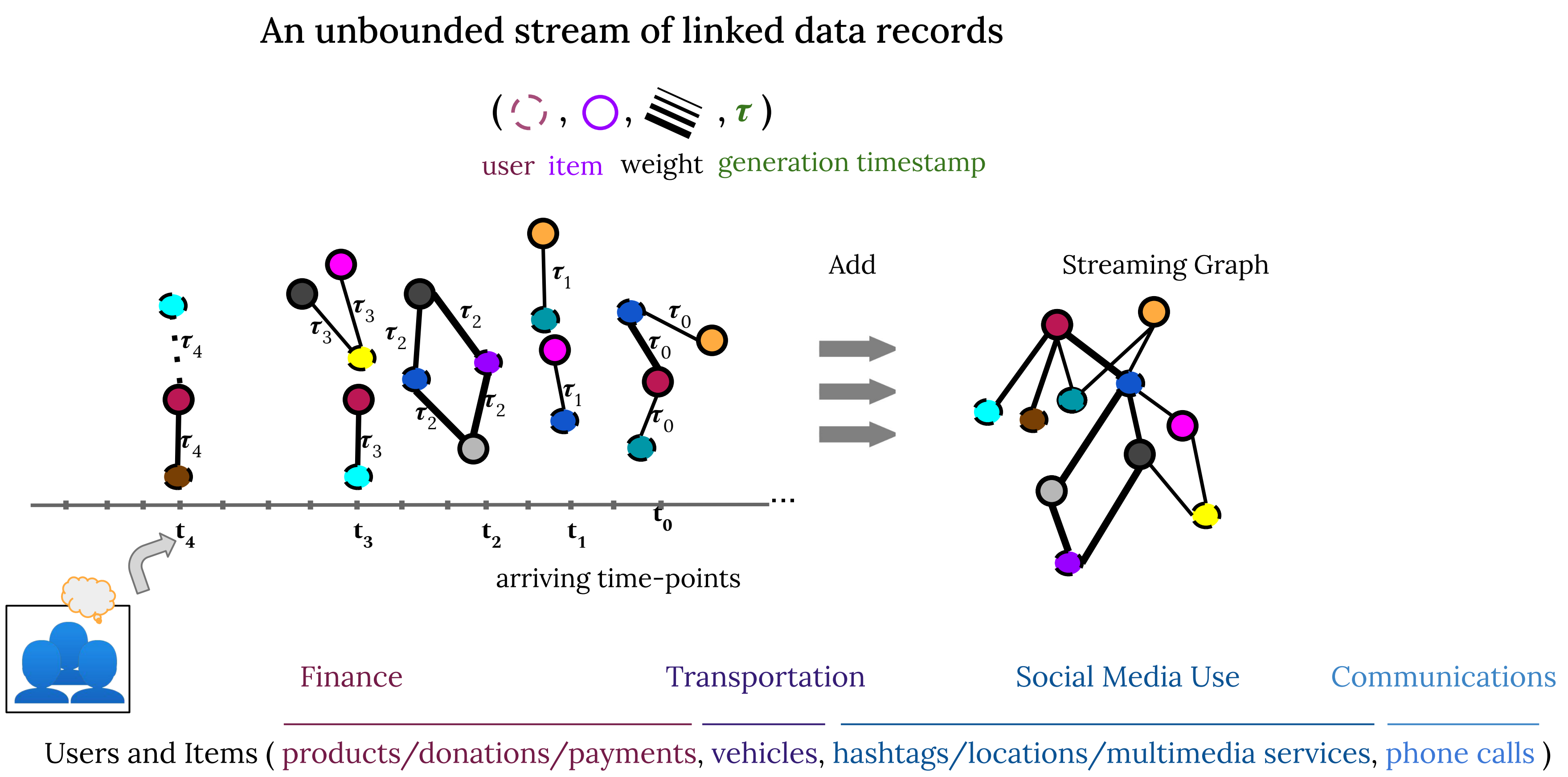
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)


An unbounded stream of linked data records
(, , , τ)
user item weight generation timestamp
Add
Streaming Graph
τ3
τ2
τ1
τ0
t3
t2
t1
t0
arriving time-points
Finance
Transportation
Social Media Use
Communications
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

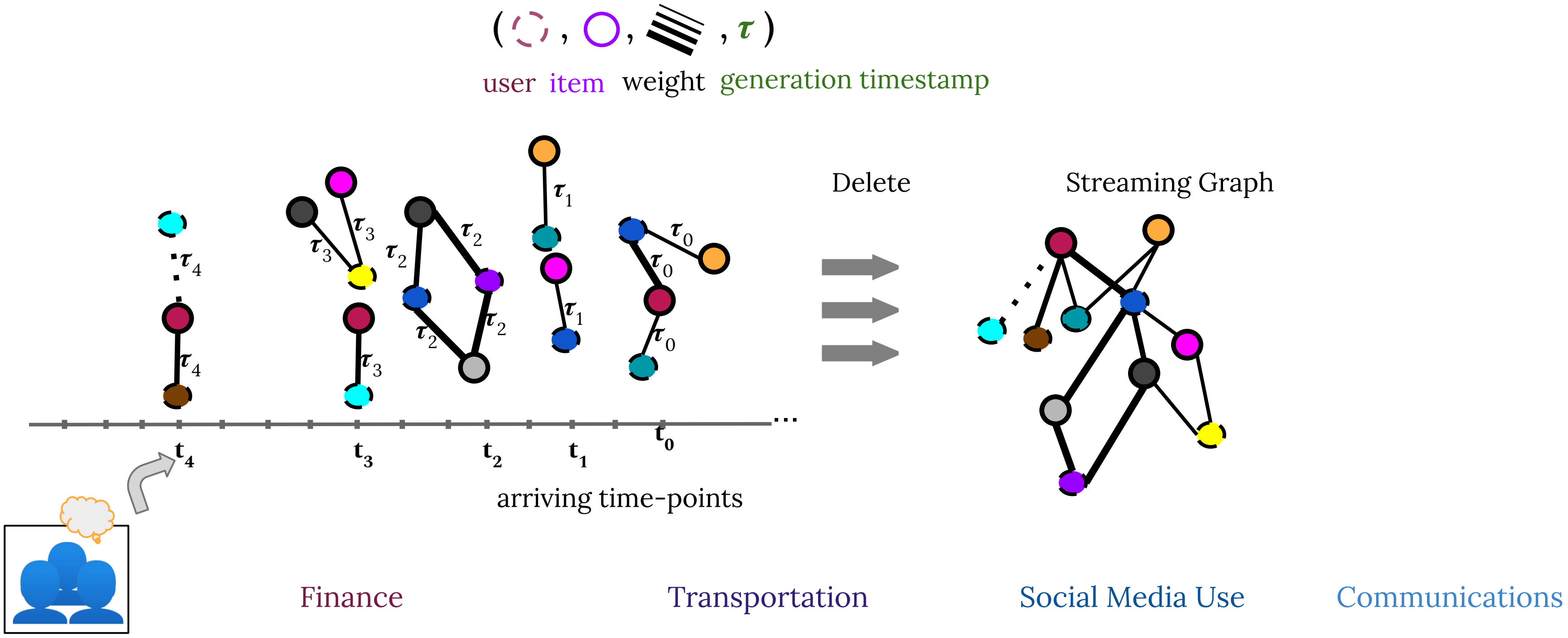

An unbounded stream of linked data records
(, , , τ)
user item weight generation timestamp
τ3
τ2
τ1
τ0
Add
Streaming Graph
t3
t2
t1
t0
...
arriving time-points
Finance
Transportation
Social Media Use
Communications
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records



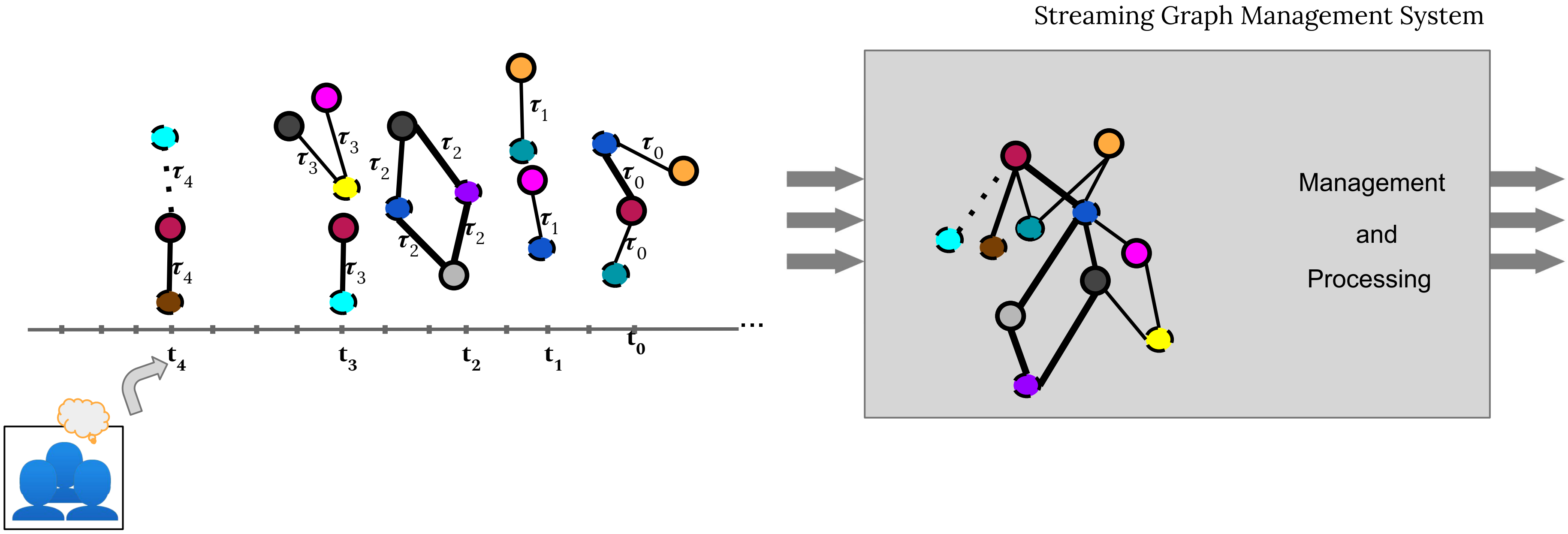

An unbounded stream of linked data records
(, , , τ)
user item weight generation timestamp
Add
Streaming Graph
τ4
τ3
τ2
τ1
τ0
t4
t3
t2
t1
t0
arriving time-points
Finance
Transportation
Social Media Use
Communications
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records



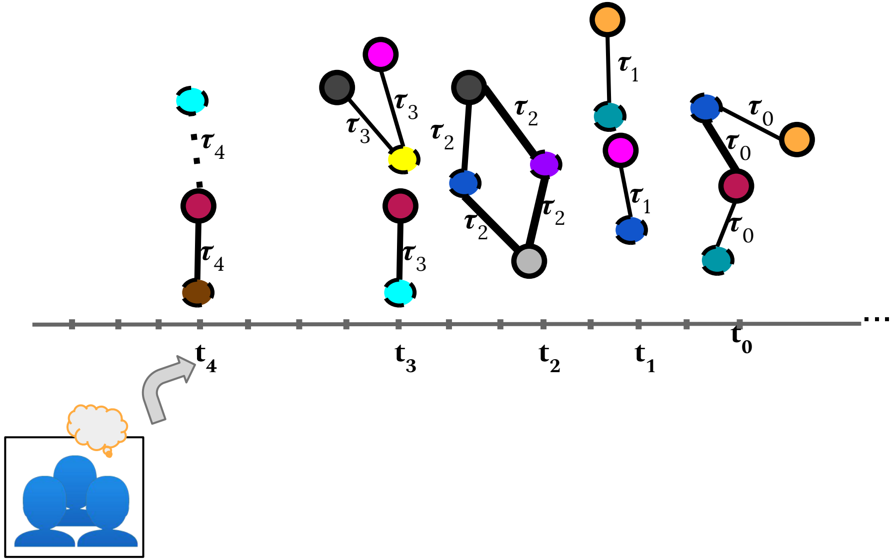
Users and Items (products/donations/payments, vehicles, hashtags/locations/multimedia services, phone calls)

An unbounded stream of linked data records

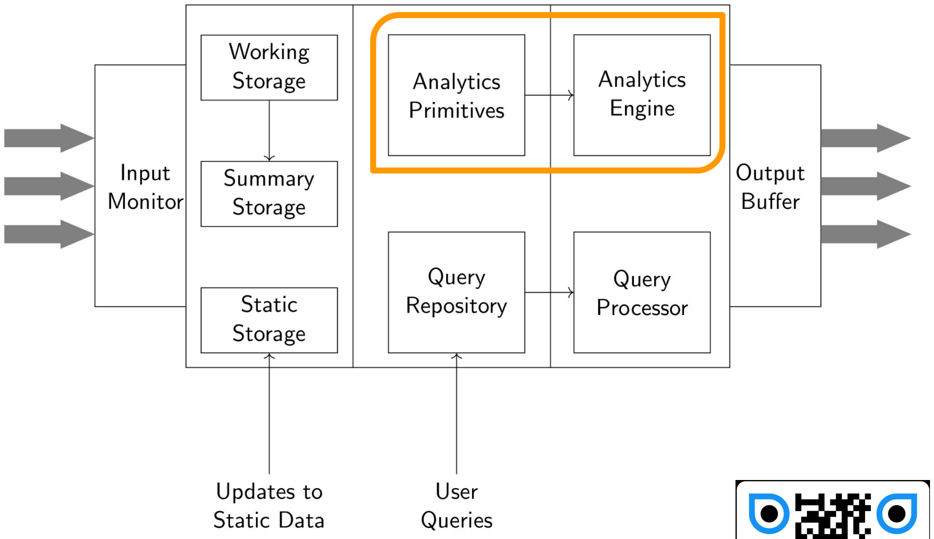


An unbounded sequence of partially ordered **Streaming Graph** Records = $<r^1, r^2, \ldots>$

$r^m = < v_i^m, v_j^m, w_{ij}^m, \tau^m >, t^m \qquad t^m \leq t^n$ for m<n



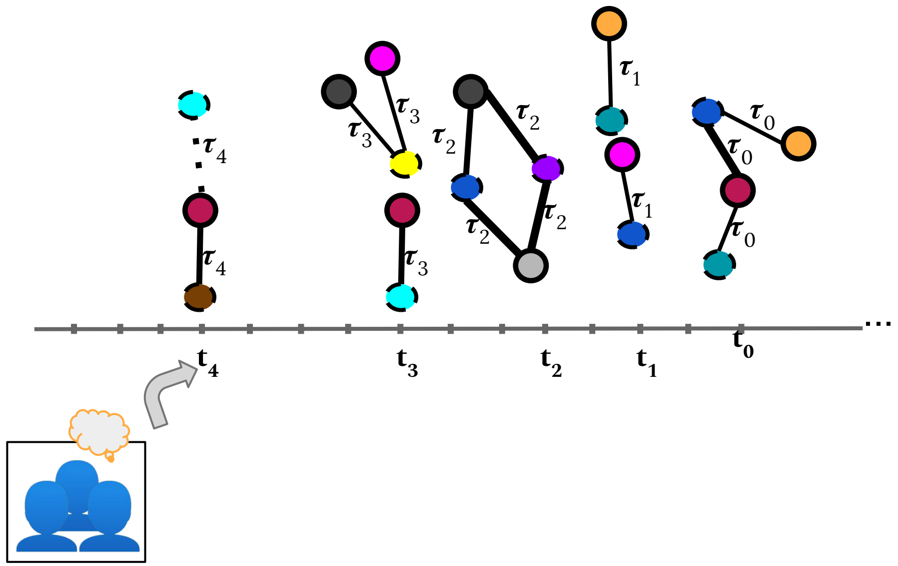
Sgraffito



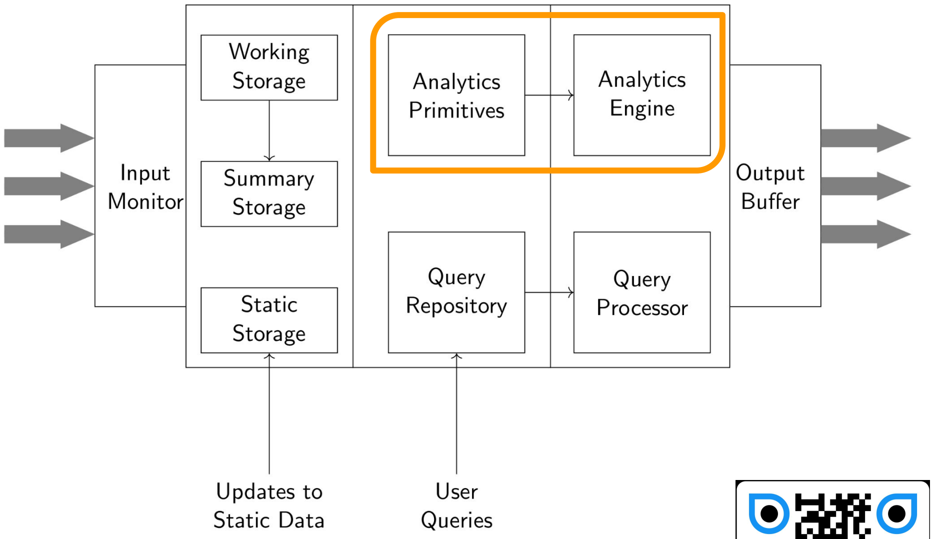
An unbounded sequence of partially ordered **Streaming Graph** Records = $< r^1, r^2, \ldots >$

$r^m = < v_i^m, v_j^m, w_{ij}^m, \tau^m >, t^m \qquad t^m \leq t^n$ for $m<n$

- Dynamic and high volume/velocity
- Reliable output



Sgraffito



Unreliable Input + Mismanagement = Unreliable Output

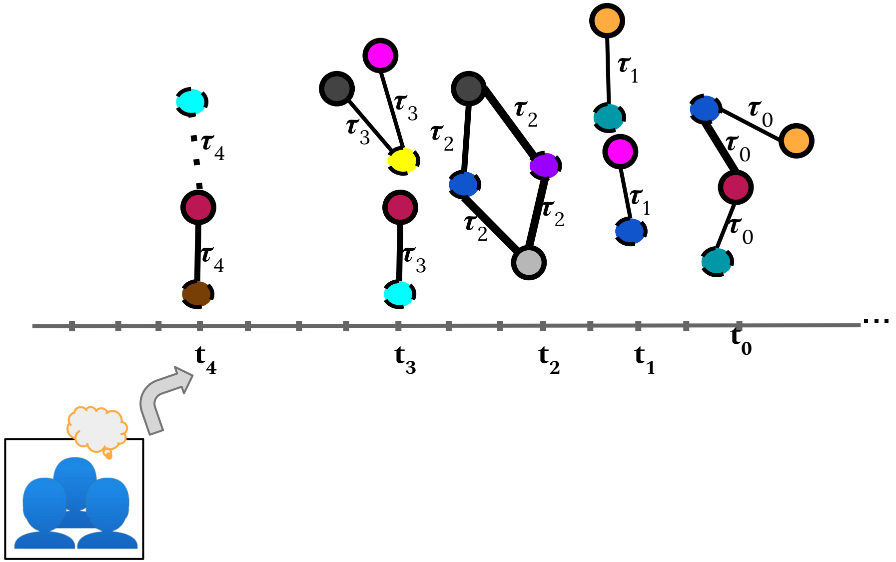
New,

Temporal,

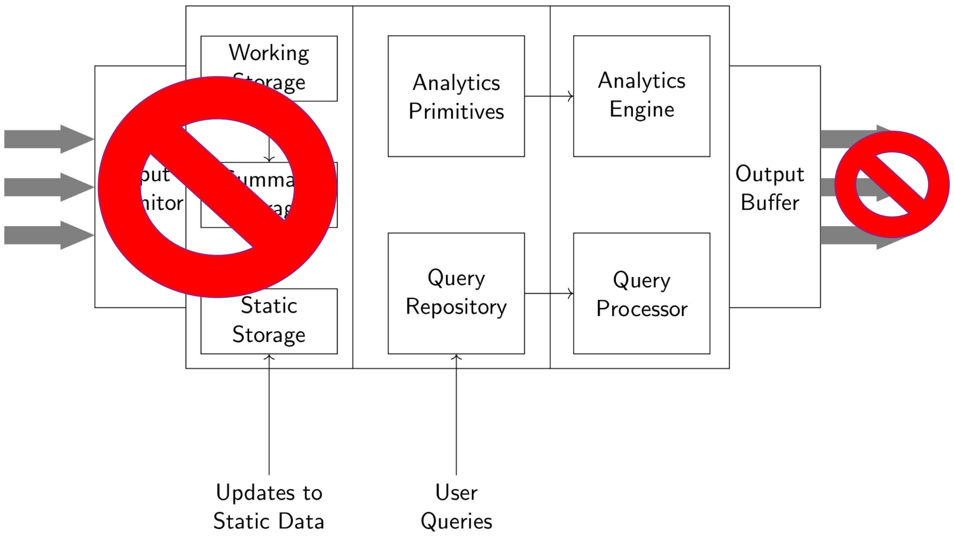
Incomplete, or

Adversely manipulated

No proper action



Sgraffito



A main cause: **Concept Drift (CD)** occurs when a change in a hidden context induces changes in a target concept.

Unreliable Input + Mismanagement = Unreliable Output

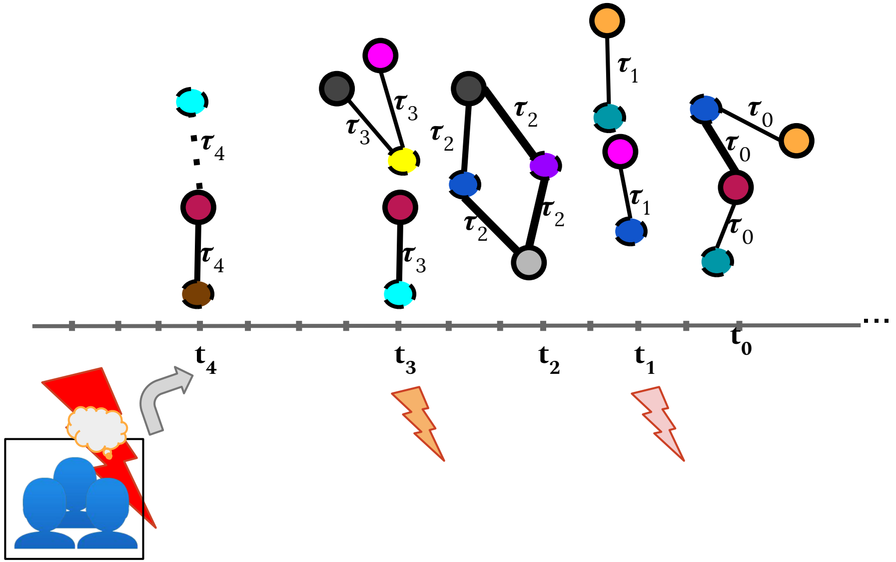
New,

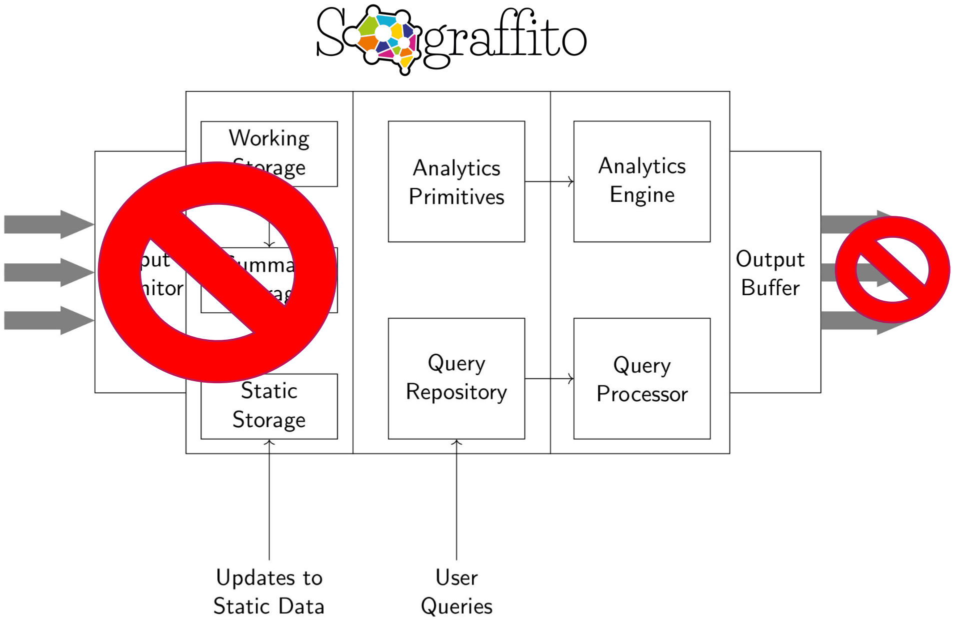
Temporal,

Incomplete, or

Adversely manipulated

No proper action



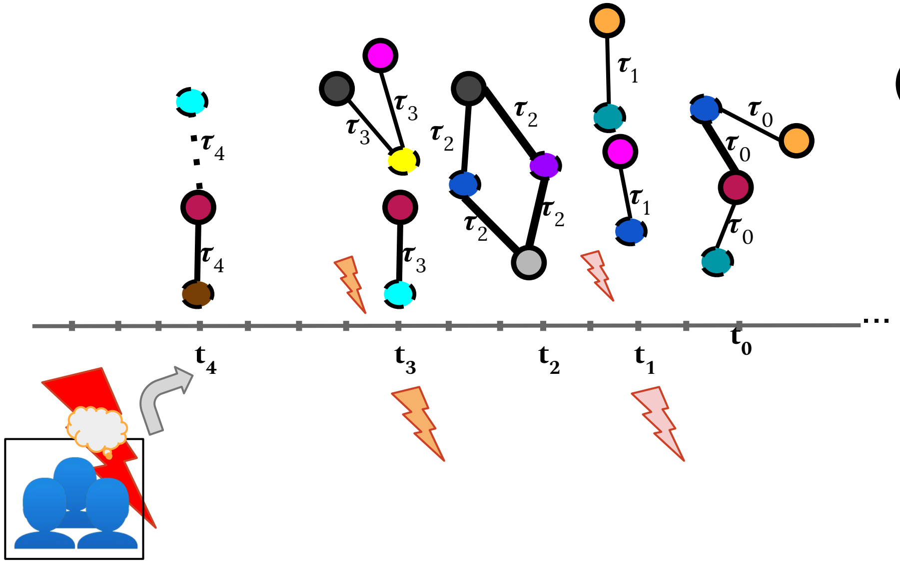


A main cause: **Concept Drift (CD)** occurs when a change in a hidden context induces changes in a target concept.

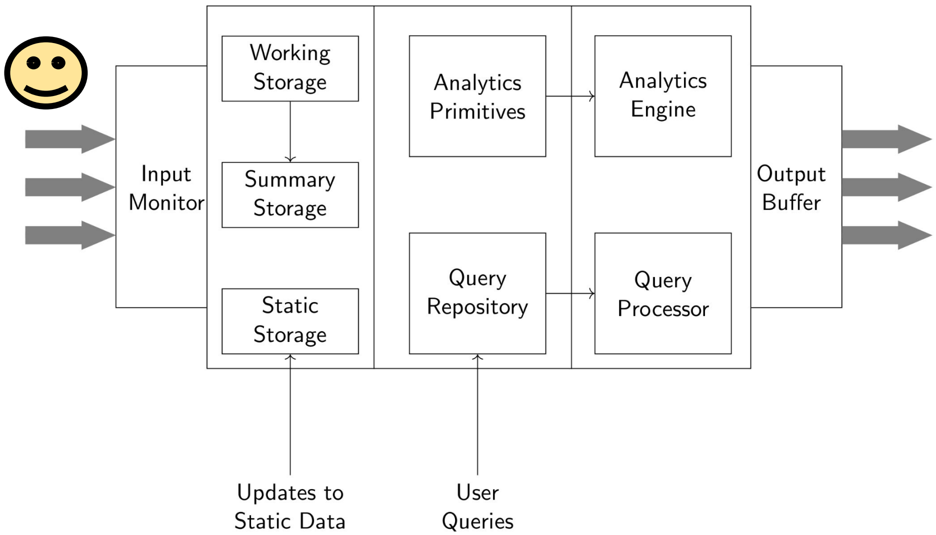
Unreliable Input + **Drift Management** = Reliable Input

New,
Temporal,
Incomplete, or
Adversely manipulated

CD Understanding
CD Detection
CD Adaptation



Sgraffito



How to detect concept drift in streaming graphs?

Existing Works on CD detection

Data

Sequence of independently generated data records

Sequence of attributed/labelled graph snapshots

Methods

Integrated drift detection and adaptation within supervised lerners

Parameterised and static techniques

CD in Streaming Graphs

Structural pattern þ refers to a quantified pattern of inter-connectivities of characteristic sub-structures (i.e. motifs) in a graph snapshot.

$þ(G_{W,t}) : G_{W,t} \rightarrow R$

Transient concept refers to a non-stable structural pattern in transient data records.

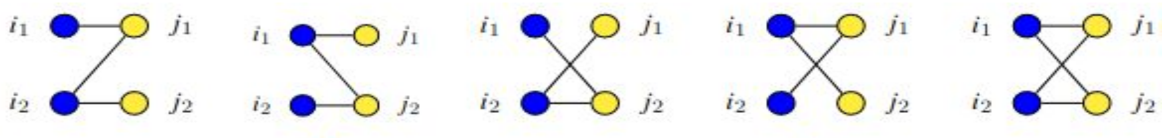
$þ(G_{W,t}) \mid \exists \ (W_1,t_1), (W_2,t_2) : \ þ(G_{W1,t1}) \neq þ(G_{W2,t2})$

Concept drift in a streaming graph is the event of a change in a transient concept over successive graph snapshots. Given a certain pattern þ, concept drift can be detected when observing at least two successive windows W_1 and W_2 corresponding to sequential time points t_1 and t_2, where t_2-$t_1 \geq 1$ and $þ(G_{W1,t1}) \neq þ(G_{W2,t2})$.

Detect changes and triggers updates
=
Concept Drift Detection

Focus on butterflies!

Goal



Caterpillar

Butterfly

A butterfly forms by adding a link to a caterpillar.

Detect changes and triggers updates

=

Concept Drift Detection

Goal

Focus on butterflies!

Stream Mining

Discover and formulate

the emergence patterns of

butterflies in streaming graphs

Detect changes and triggers updates
=
Concept Drift Detection

Focus on butterflies!

Goal

Graph Analytics

sGrapp: A framework for
Streaming **Gr**aph Butterfly **App**roximation

Stream Mining

Discover and formulate
the emergence patterns of
butterflies in streaming graphs

Mining Butterflies in Streaming Graphs

Detect changes and triggers updates
=
Concept Drift Detection

Goal

Focus on butterflies!

Graph Analytics

sGrapp: A framework for
Streaming **Gr**aph Butterfly **App**roximation

Stream Mining

Discover and formulate
the emergence patterns of
butterflies in streaming graphs

Explainable Modelling

sGrow: A framework for
modelling **S**treaming Graph **Grow**th

Detect changes and triggers updates
=
Concept Drift Detection

Focus on butterflies!

Goal

Graph Analytics

sGrapp: A framework for **S**treaming **Gr**aph Butterfly **App**roximation

sGradd: a framework for **S**treaming **Gra**ph **D**rift **D**etection

Stream Mining

Discover and formulate the emergence patterns of butterflies in streaming graphs

Explainable Modelling

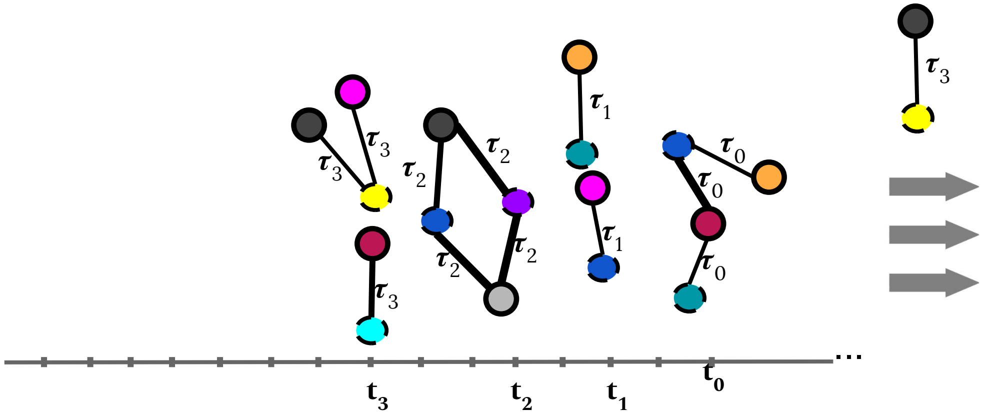
sGrow: A framework for modelling **S**treaming Graph **Grow**th

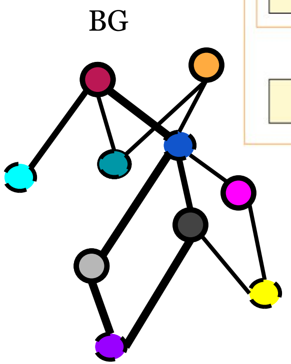
Mining Butterflies in Streaming Graphs

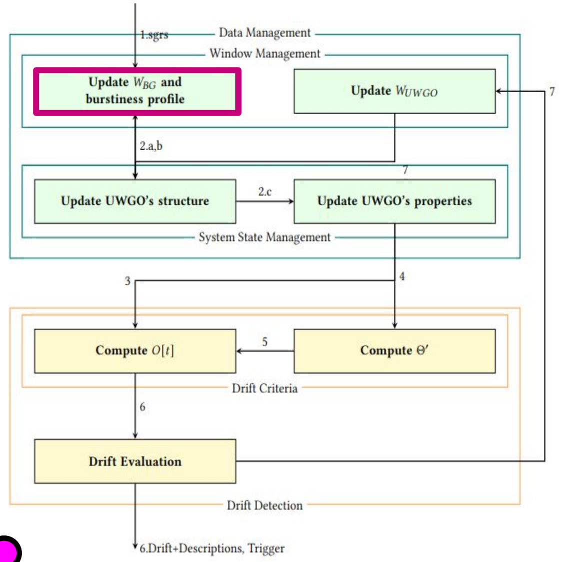
Aida Sheshbolouki (2023). UWspace

sGradd: a framework for

Streaming **Gra**ph **D**rift **D**etection

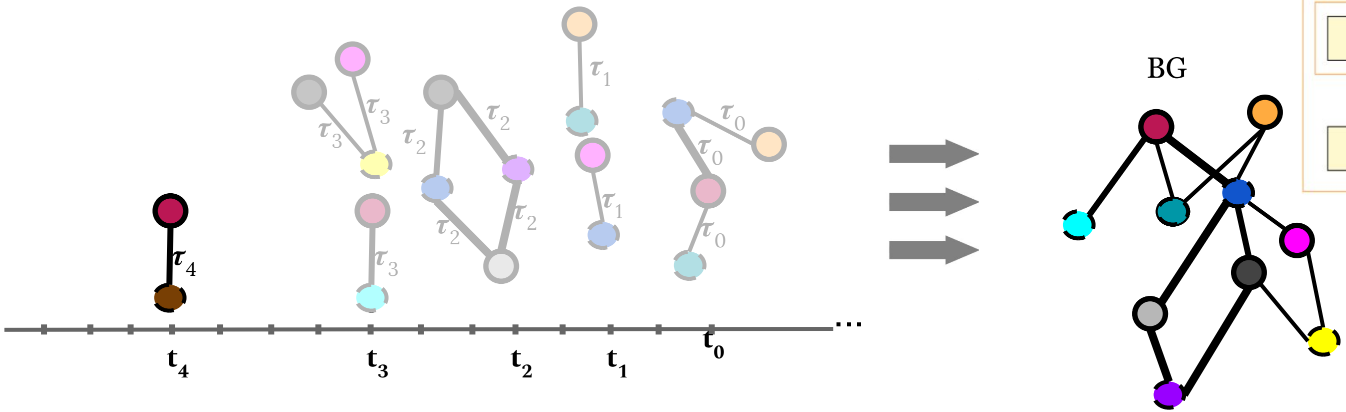


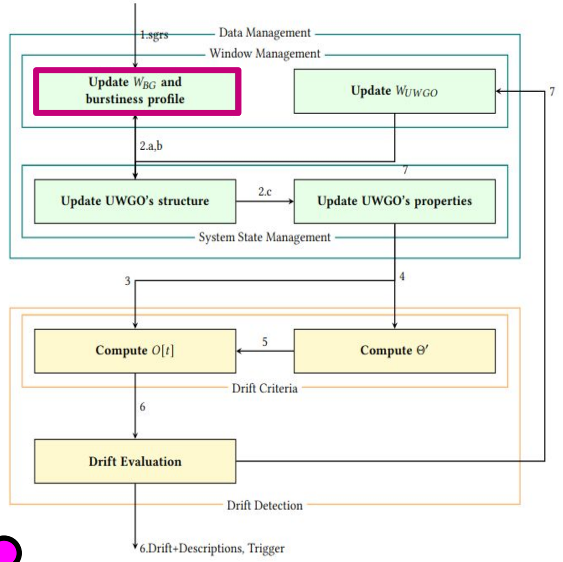




sGradd: a framework for

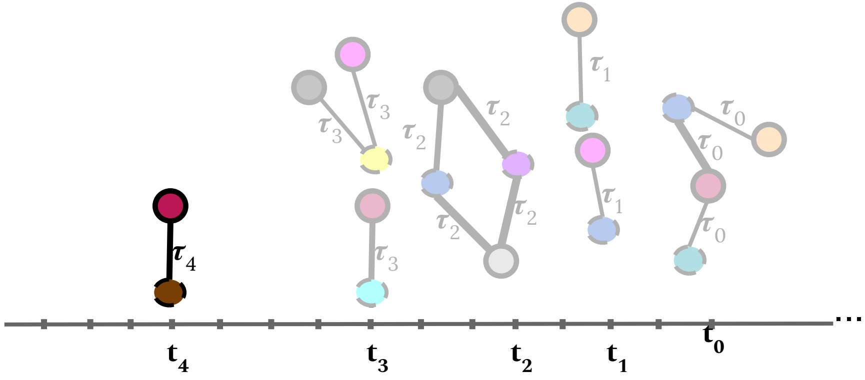
Streaming **Gra**ph **D**rift **D**etection

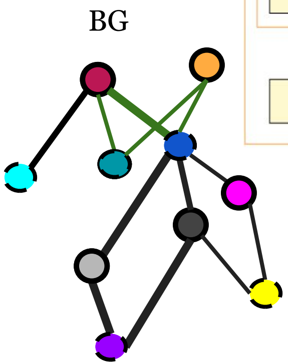


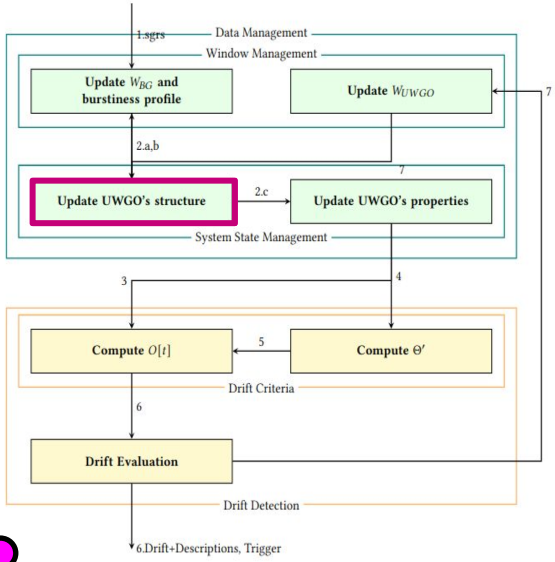


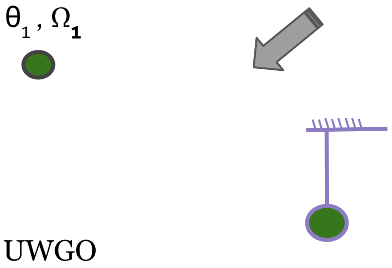
sGradd: a framework for

Streaming **Gra**ph **D**rift **D**etection









For each butterfly $\bowtie_v$ in W_{BG}

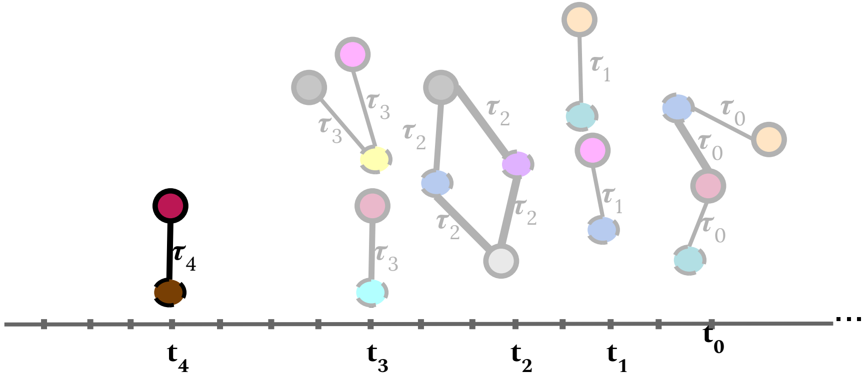
Create a vertex v with $\theta_v=0$, $\Omega_v=0$

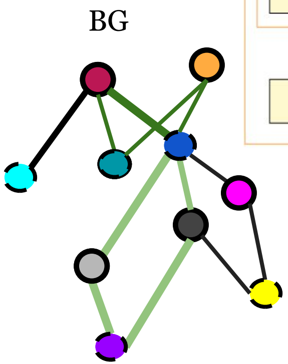
Connect v to n with $\bowtie_n \cap \bowtie_v \neq \emptyset$

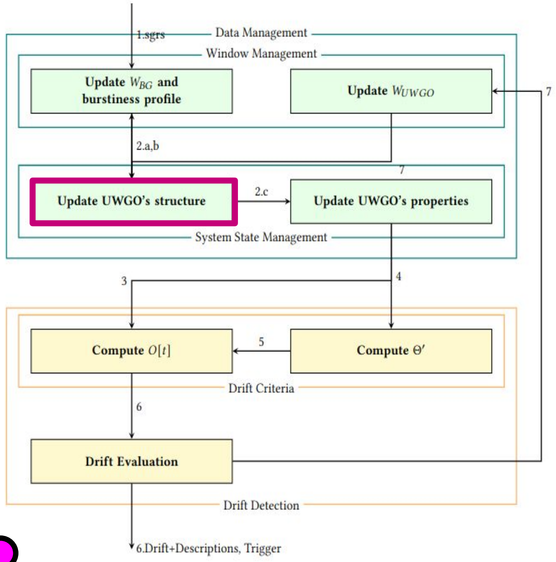
$W_{vn} = |N(v) \cup N(n)|$

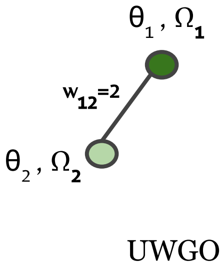
sGradd: a framework for

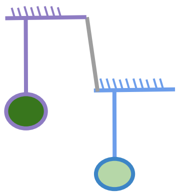
Streaming **Gra**ph **D**rift **D**etection











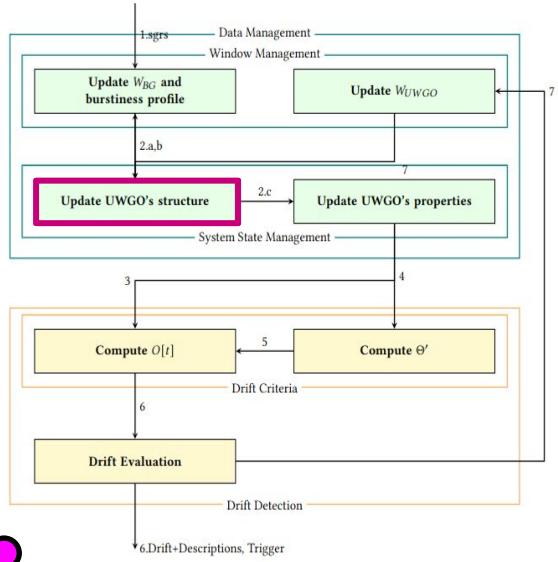
For each butterfly $\bowtie_v$ in W_{BG}

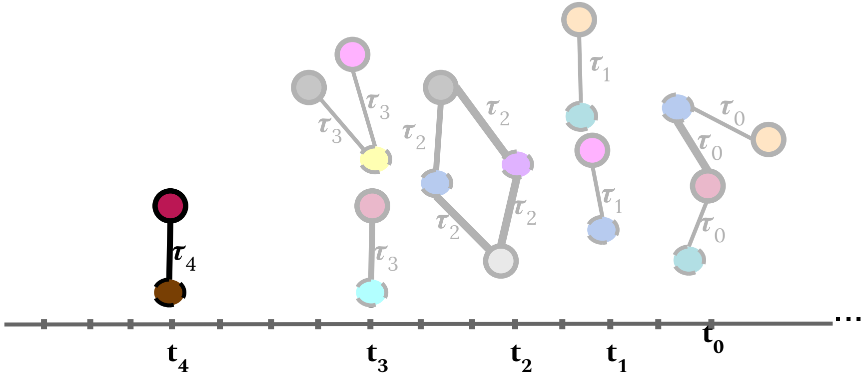
Create a vertex v with $\theta_v=0$, $\Omega_v=0$

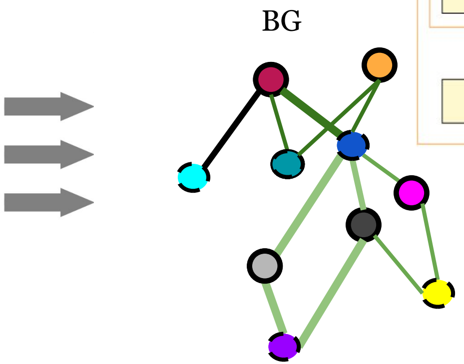
Connect v to n with $\bowtie_n \cap \bowtie_v \neq \emptyset$

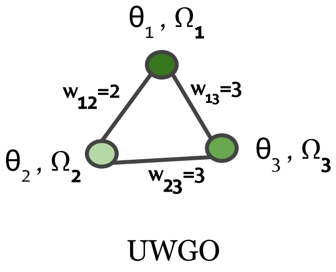
$W_{vn} = |N(v) \cup N(n)|$

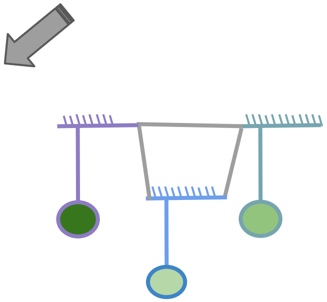
sGradd: a framework for **S**treaming **Gra**ph **D**rift **D**etection











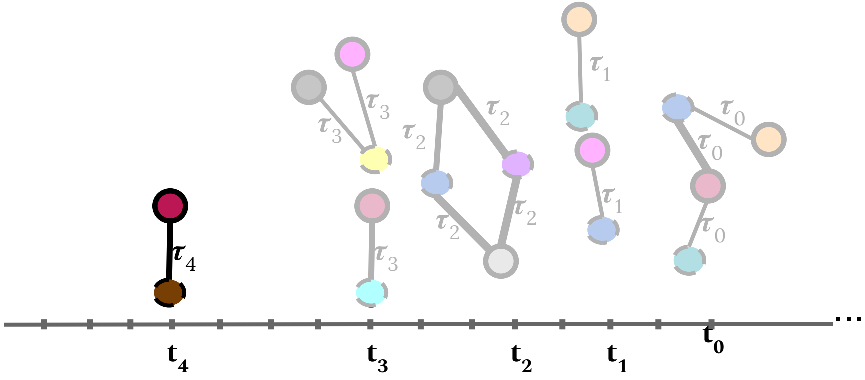
For each butterfly $\bowtie_v$ in W_{BG}

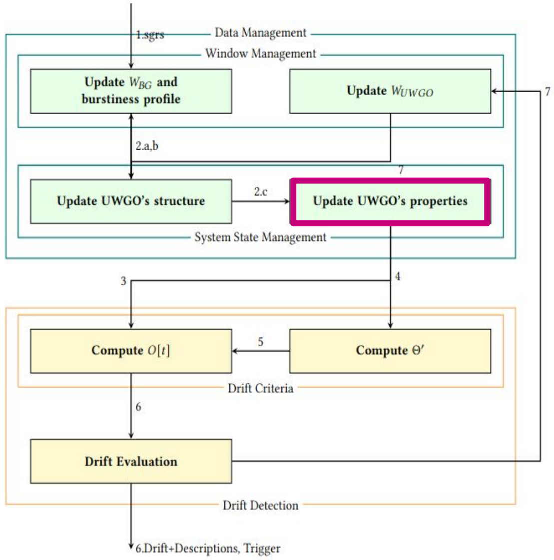
Create a vertex v with $\theta_v=0$, $\Omega_v=0$

Connect v to n with $\bowtie_n \cap \bowtie_v \neq \emptyset$

$W_{vn} = |N(v) \cup N(n)|$

sGradd: a framework for **S**treaming **Gra**ph **D**rift **D**etection

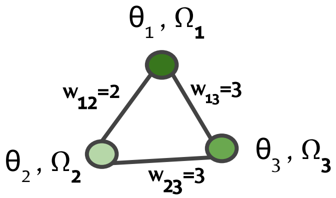




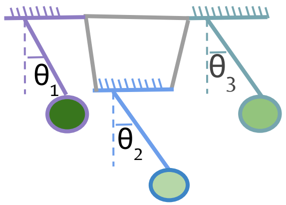
For each vertex v in W_{UWGO}

$\theta_v := (\Sigma_{n \in N(v)} \text{HashCode}(n)) \ \% \ 2\pi$

$\Omega_v :=$ sample from a distribution

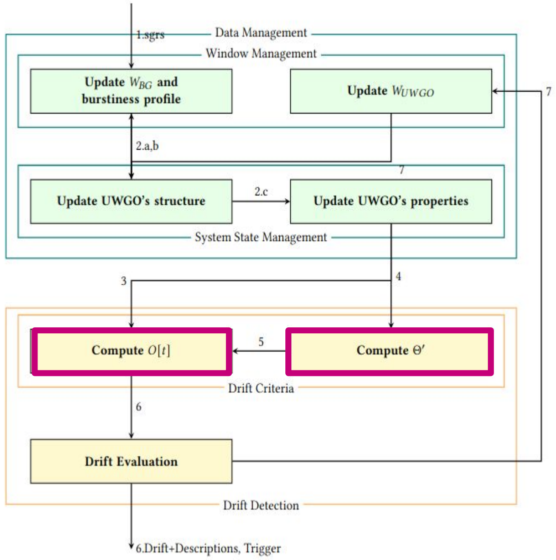


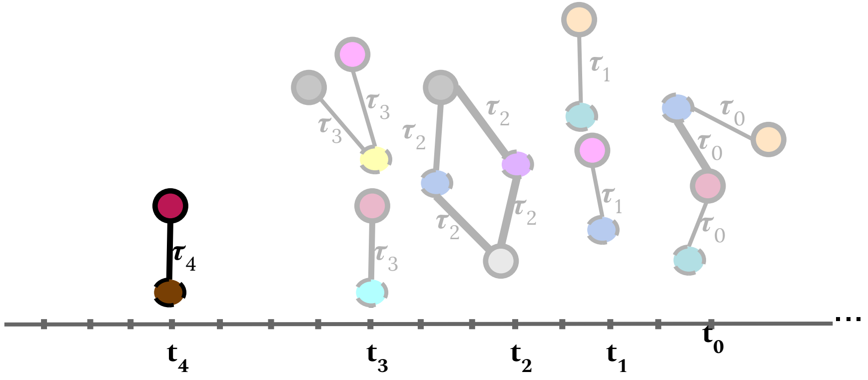
UWGO

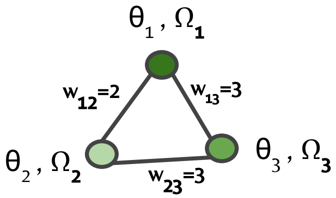


sGradd: a framework for

Streaming **Gra**ph **D**rift **D**etection







UWGO

Global Phase Synchronization $0<\mathbf{O[t]}<1$

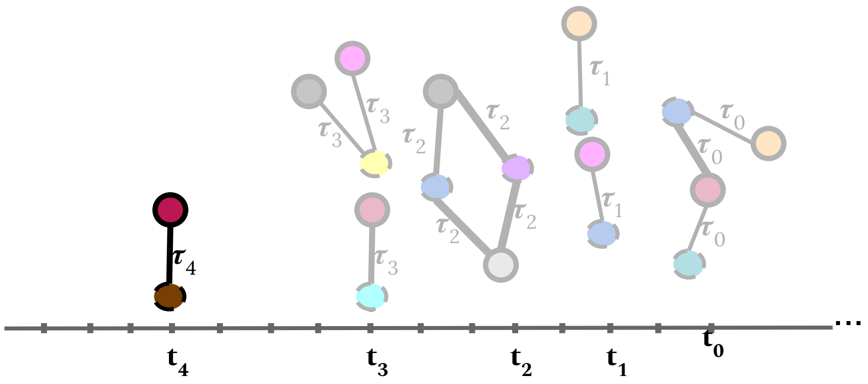
$$O_1[t_4] = \left((\Sigma_{v\in V} \sin \theta_v)^2 + (\Sigma_{v\in V} \cos \theta_v)^2 \right)^{1/2} / |V|$$

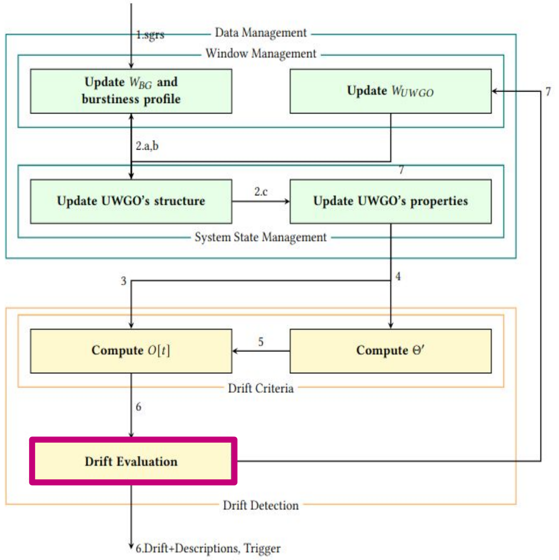
Kuramoto: $d\theta_v/dt = \Omega_v + \Sigma_{n\in N(v)} w_{vn} \sin(\theta_v - \theta_n)$

$\forall\ v\in V,\ \Theta^2_v \leftarrow$ Runge Kutta

$$O_2[t_4] = \left((\Sigma_{v\in V} \sin \theta^2_v)^2 + (\Sigma_{v\in V} \cos \theta^2_v)^2 \right)^{1/2} / |V|$$

sGradd: a framework for **S**treaming **Gra**ph **D**rift **D**etection



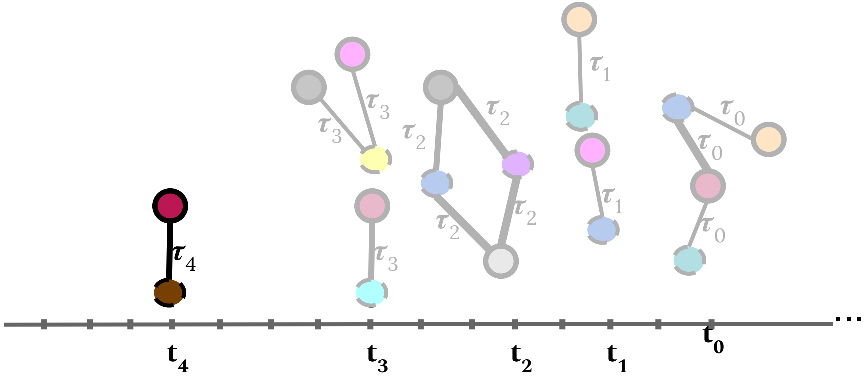


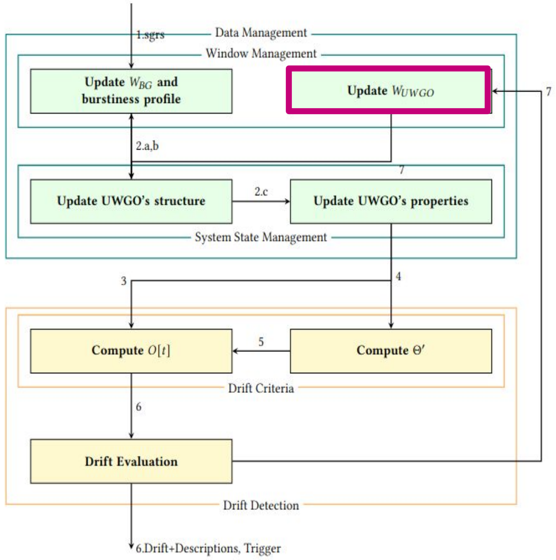
Signal a drift when

A local maximum/minimum is observed in O2, while O1 remains steadily fixed.

sGradd: a framework for

Streaming **Gra**ph **D**rift **D**etection

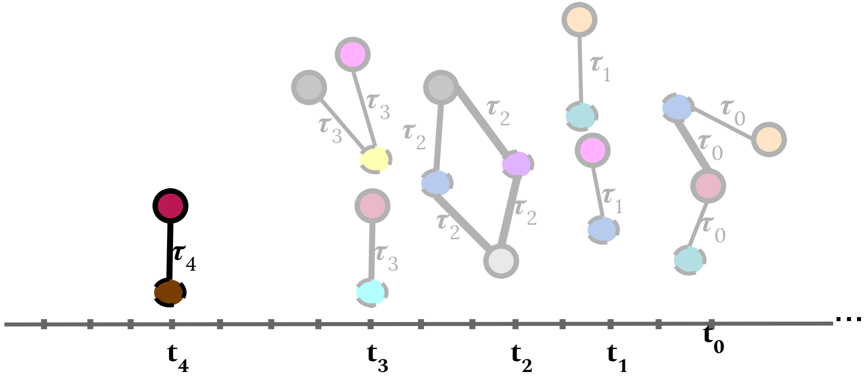


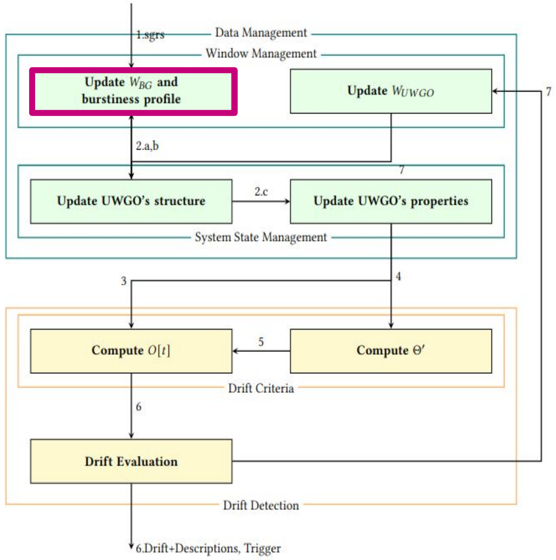


If stream is bursty,

Remove random vertices from W_{UWGO}

sGradd: a framework for Streaming Graph Drift Detection





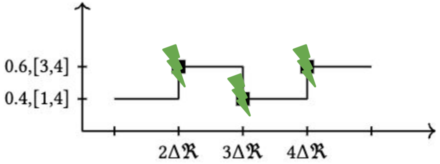
Performance Evaluation

Data: *20 labelled stream with* 10^6 *records* ← *sGrow* (G_0 , M , β , $\boldsymbol{\rho}$, $[\mathbf{L}_{min}\mathbf{L}_{max}]$) + *CD labels*

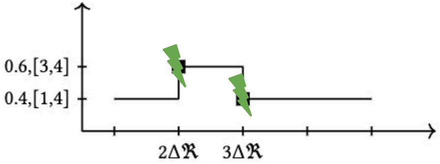
2 CD *patterns*

2 CD *intervals:* $\Delta\mathfrak{R}=10^5$, 2×10^5

5 *instances per pattern per interval*



(a) Three changes for gradual CD



(b) Two changes for recurring CD

Metrics: *True/False/Miss detection rates, Detection Delay - Average over 50 executions*

***Computing Setup*:** 15.6GB *native memory*, Intel Core i7 - 2.6GHz*8 processor, Java

Algorithms: Three variants of CD detection module in sGradd

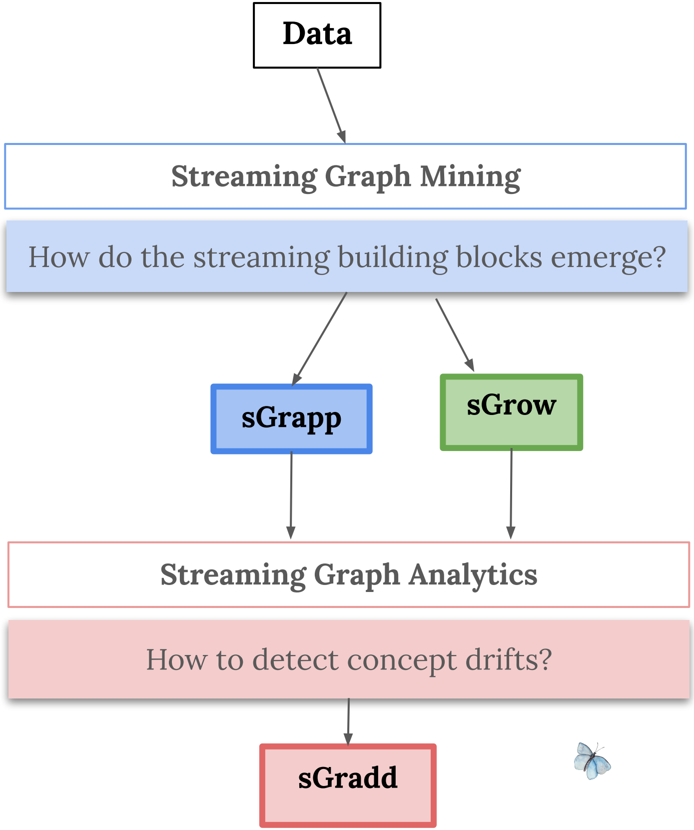
Accuracy of sGradd over streams with 2×10^5 CD interval

Latency of sGradd over streams with 2×10^5 CD interval

Accuracy and Latency of sGradd over streams with 2×10^5 CD interval

Dependency of local variables and their impact on the performance

+ Improving accuracy
+ Zero false alerts
+ Average detection delay in [9, 373] & [5, 310] (s) for close and far drift intervals



A composable framework for streaming graph drift detection which

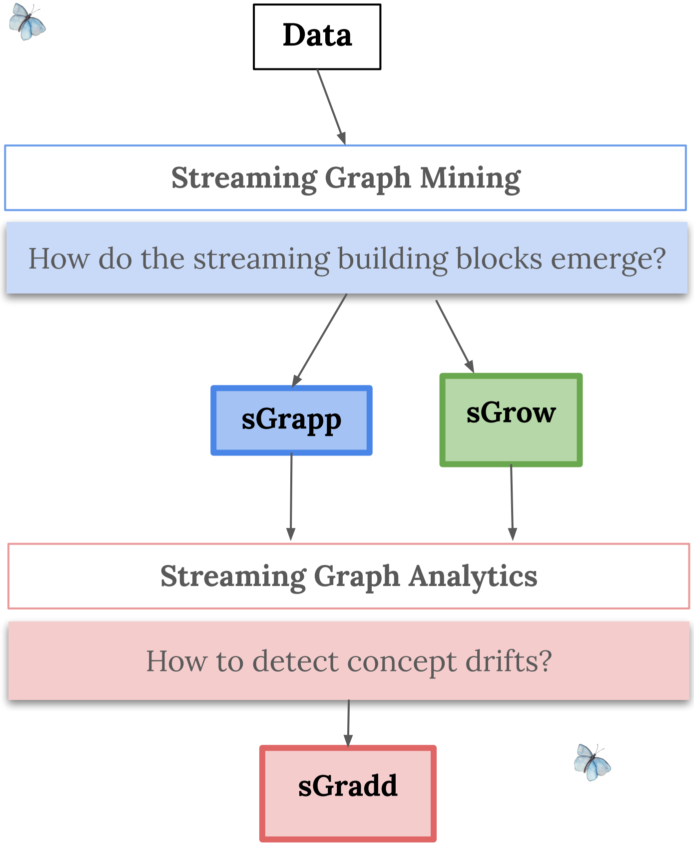
- supports any analytics (supervised or unsupervised),
- incurs low overhead, while accurately detecting the drifts,
- detects various drift types,
- explains the detected drifts' time and location,
- detects drifts without supervision,
- adapts to the streaming rate, and
- does not require input thresholds, prototype parameters, and graph attributes.

A performance evaluation which

- simulates concept drift realistically
- examines the accuracy and latency of detections effectively

Thank you

aida.sheshbolouki@uwaterloo.ca

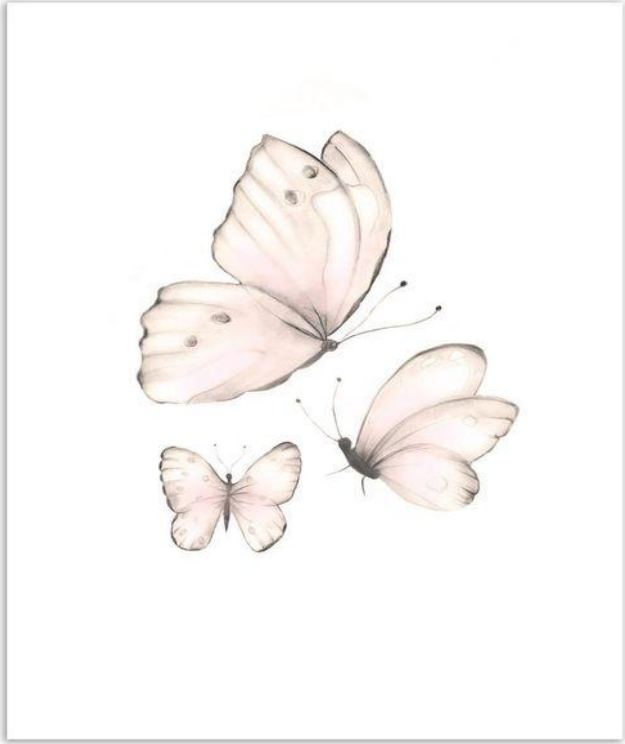


A composable framework for streaming graph drift detection which

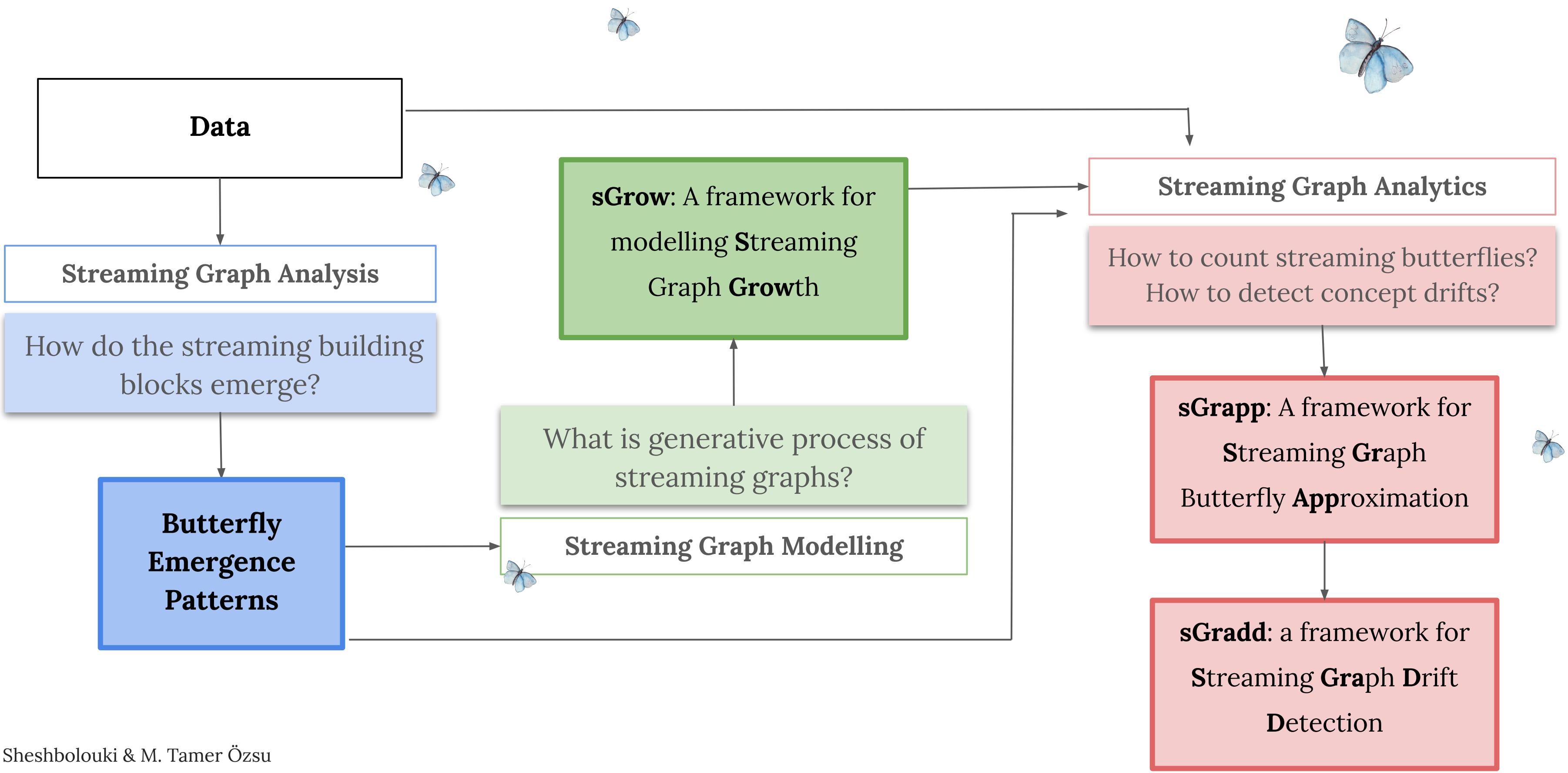
- supports any analytics (supervised or unsupervised),
- incurs low overhead, while accurately detecting the drifts,
- detects various drift types,
- explains the detected drifts' time and location,
- detects drifts without supervision,
- adapts to the streaming rate, and
- does not require input thresholds, prototype parameters, and graph attributes.

A performance evaluation which

- simulates concept drift realistically
- examines the accuracy and latency of detections effectively



Supporting Material

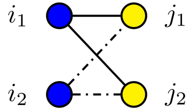


Aida Sheshbolouki & M. Tamer Özsu
"*sGrapp: Butterfly approximation in streaming graphs.*" ACM Transactions on Knowledge Discovery from Data 16.4 (2022): 1-43.
"*sGrow: Explaining the Scale-Invariant Strength Assortativity of Streaming Butterflies.*" ACM Transactions on the Web (2022).

Stream Mining

Discover and formulate the emergence patterns of butterflies in streaming graphs

Statistical Graph Analysis

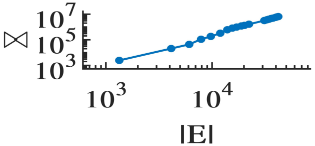


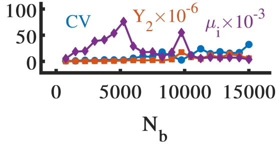
Tools and techniques:

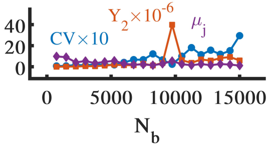
- A butterfly counting algorithm for static graph snapshots
- A metric for evaluating data distribution within/across streams
- Window management techniques for stream mining

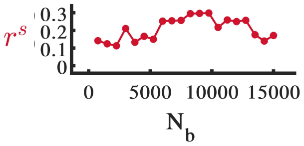
Butterfly emergence patterns:

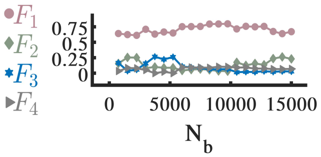
- *Butterfly Densification Power Law*: Bursty butterfly formation and its origins
- *Scale-Invariant Strength Assortativity of Butterflies*: Butterfly mixing patterns

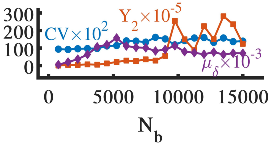












Scale-Invariant Strength Assortativity of Butterflies

Butterfly densification

The number of butterflies grows over time and at each time point it is a super-linear function of the number of edges.

Diversification of strengths

The distribution of strengths gets broader and more diverse over time.

Steady strength assortativity

The strength assortativity coefficient is fixed at a positive value over time due to the fixed-shaped yet growing distribution of strength-difference.

While the new high-strength vertices with low strength-neighbors form butterflies and the variance of strength differences increases with the arrival new vertices and edges, the majority of butterflies are formed by vertices with similar strength. We refer to this phenomenon as the scale-invariant strength assortativity.

Graph Analytics

sGrapp: A framework for **S**treaming **Gr**aph Butterfly **App**roximation

Total butterflies until the end of window W_k

Butterflies in window W_k

$$B_k = B_{k-1} + B_{W_k} + B_{\text{inter window}}$$

$$B_0 = B_{W0}$$

Approximating inter-window butterflies using BPL

Data-driven approach for subgraph listing

Benefits study of cohesion for

- **Marketing.** Finding groups of similar entities and recommendations
- **Information management/monitoring.** Analyzing information propagation
- **Health care.** Dynamic phenomena such as epidemic spreading

Incremental processing powered by:

- Butterfly densification power law (BPL) for approximation
- Identifying performance bottlenecks: *inter-window* butterflies
- Window management adaptive to streaming rate

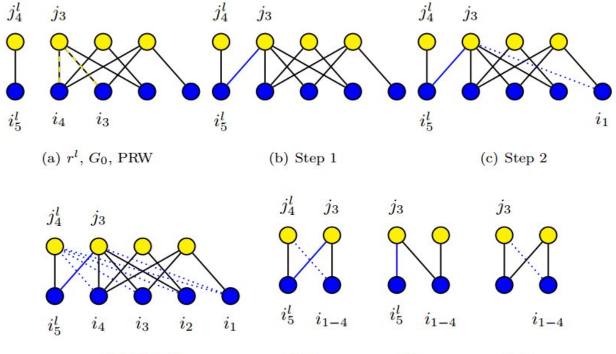
Performance:

- Simultaneous effectiveness and efficiency
- 160x higher throughput and 0.02x lower estimation error than baselines
    - average window errors <0.05 and 0.14 in streams with uniform and non-uniform temporal distributions
    - processing throughput of 1.5×10^6 data record per second

Aida Sheshbolouki & M. Tamer Özsu
"*sGrapp: Butterfly approximation in streaming graphs.*"
ACM Transactions on Knowledge Discovery from Data 16.4 (2022): 1-43.

Explainable Modelling

sGrow: A framework for modelling **S**treaming Graph **Grow**th



(a) r^l, G_0, PRW (b) Step 1 (c) Step 2 (d) Step 3 (e) (f) (g)

Micro-mechanisms for explaining the discovered patterns

A graph traversal approach: preferential random walk

- Selecting nodes according to their weighted degree (strength preferential selection)
- Breadth-First and Depth-First traversals
- Dynamic and random number of visited nodes

Realistic streaming record generation

- Inactivity gaps
- Timestamp assignment
- Evolving streaming rate
- Local and unbounded graph updates

Probabilistic connections

- Random nodes
- Neighbour copying

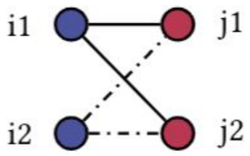
Performance:

- Robust and efficient realization of streaming growth patterns
- Independent of initial conditions, scale and temporal characteristics, and model configurations.

Aida Sheshbolouki & M. Tamer Özsu
"*sGrow: Explaining the Scale-Invariant Strength Assortativity of Streaming Butterflies.*"
ACM Transactions on the Web (2022).

Listing the butterflies in sequence of burst-based graph snapshots of the stream:

sGrapp's core algorithm



Algorithm 1: countButterflies(G)

Input: $G = \langle V_i \cup V_j, E_{ij} \rangle$, static graph

Output: B_G, The number of butterflies in G

$Butterflies \leftarrow \emptyset$ // An empty hashSet of quadruples

$jNeighbors \leftarrow \emptyset$ // An empty Set

$vi2s \leftarrow \emptyset$ // An empty Set

/* loop over $v_{i1} \in V_i$ if $K_i < K_j$, otherwise loop over $v_{j1} \in V_j$ */

for $v_{i1} \in V_i$ **do**

  $jNeighbors \leftarrow N_{v_{i1}}$ // j-neighbors of vertex v_{i1}

  for $index1 \in [1, size(jNeighbors)]$ **do**

    $v_{j1} \leftarrow jNeighbors[index1]$

    for $index2 \in [index1 + 1, size(jNeighbors)]$ **do**

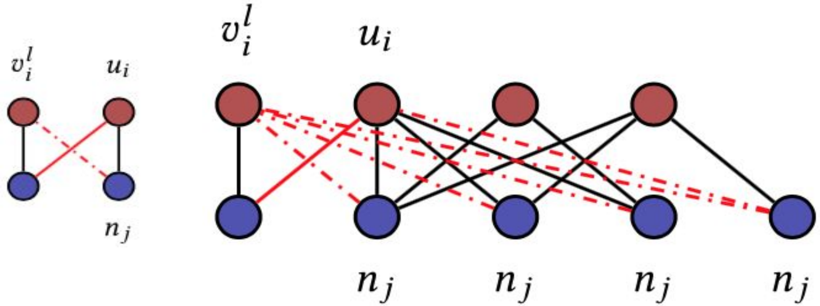
      $v_{j2} \leftarrow jNeighbors[index2]$

      $vi2s \leftarrow N_{v_{j1}} \cap N_{v_{j2}}$ // common i-neighbors

      $Butterflies.add([v_{i1}, v_{j1}, v_{i2}, v_{j2}])$

$B_G \leftarrow size(Butterflies)$

Explaining the Scale-invariant strength assortativity of butterflies: **sGrow's algorithms**



Algorithm 4: Add Burst

Function $addBurst(v_i^l, v_j^l, PRW_i, G, \mathfrak{R}, \rho)$
  for ***each*** $u_i \in PRW_i$ **do**
    $\omega' \leftarrow$ a random integer in $[1, 5]$
    Add a new sgr $\langle u_i, v_j^l, \omega', v_j^l.\tau \rangle$ to $\mathfrak{R}$ and G
    if *coin*(ρ) *is Head* **then**
      $z_j \leftarrow$ Select a random j-vertex
      $\omega' \leftarrow$ a random integer in $[1, 5]$
      Add a new sgr $\langle u_i, z_j, \omega', Min(u_i.\tau, z_j.\tau) \rangle$ to $\mathfrak{R}$ and G
    for ***each*** $n_j \in N_j(u_i)$ **do** // in bursty streams with $b > 2$
      if *coin*(ρ) *is Head* **then**
        $\omega' \leftarrow$ a random integer in $[1, 5]$
        Add a new sgr $\langle v_i^l, n_j, \omega', n_j.\tau \rangle$ to $\mathfrak{R}$ and G

$|N_j(u_i)|$ caterpillars → butterflies

Algorithm 1: Graph Model

Input: ρ: connection probability, M: number of isolated edges, β: slide parameter, $[L_{min}, L_{max}]$: PRW's length range
Output: $\mathfrak{R}$, sequence of streaming graph records

$G \leftarrow G_0 = (V_0, E_0)$ // computational graph
$\mathfrak{R} \leftarrow E_0$ // sequence of sgrs
$\tau \leftarrow$ 1+ last timestamp in G_0 // timestamp
$t \leftarrow 0$ // timestep
$W^b \leftarrow$ first timestamp in G_0 // sliding window's beginning border
while *true* **do**
  $t \leftarrow t + 1$
  Add $m \in [0, M)$ new sgrs $r^{l=0,..,m}$ to $\mathfrak{R}$ and G
  for ***each*** $r^{l=0,..,m} = \langle v_i^l, v_j^l, \omega_{ij}^l, \tau \rangle$ **do**
    $\omega \leftarrow$ a random integer in $[-1, 5]$
    switch ω **do**
      case *-1* **do**
        Remove any edge between v_i^l and v_j^l from $\mathfrak{R}$ and G.
      case *0* **do**
        No operation
      otherwise do
        $init_j \leftarrow SPS(V_j)$
        $L \leftarrow$ a random integer in $[L_{min}, L_{max}]$
        $(PRW_i, PRW_j) \leftarrow \text{PRW}(init_j, false, G, L)$
        $addBurst(v_i^l, v_j^l, PRW_i, G, \mathfrak{R}, \rho)$
        $addBurst(v_i^l, v_j^l, PRW_j, G, \mathfrak{R}, \rho)$
        $\tau \leftarrow \tau + \frac{(\omega'-5)(\omega'-4)(\omega'-3)}{2}$
  Remove any newly added vertex v_i^l and v_j^l with less than 2 neighbors from G
  $\tau \leftarrow \tau + 1$
  $W^b \leftarrow W^b + \beta$
  if $t = \beta$ **then**
    Remove any edge with timestamp less than W^b from G
    $t \leftarrow 0$